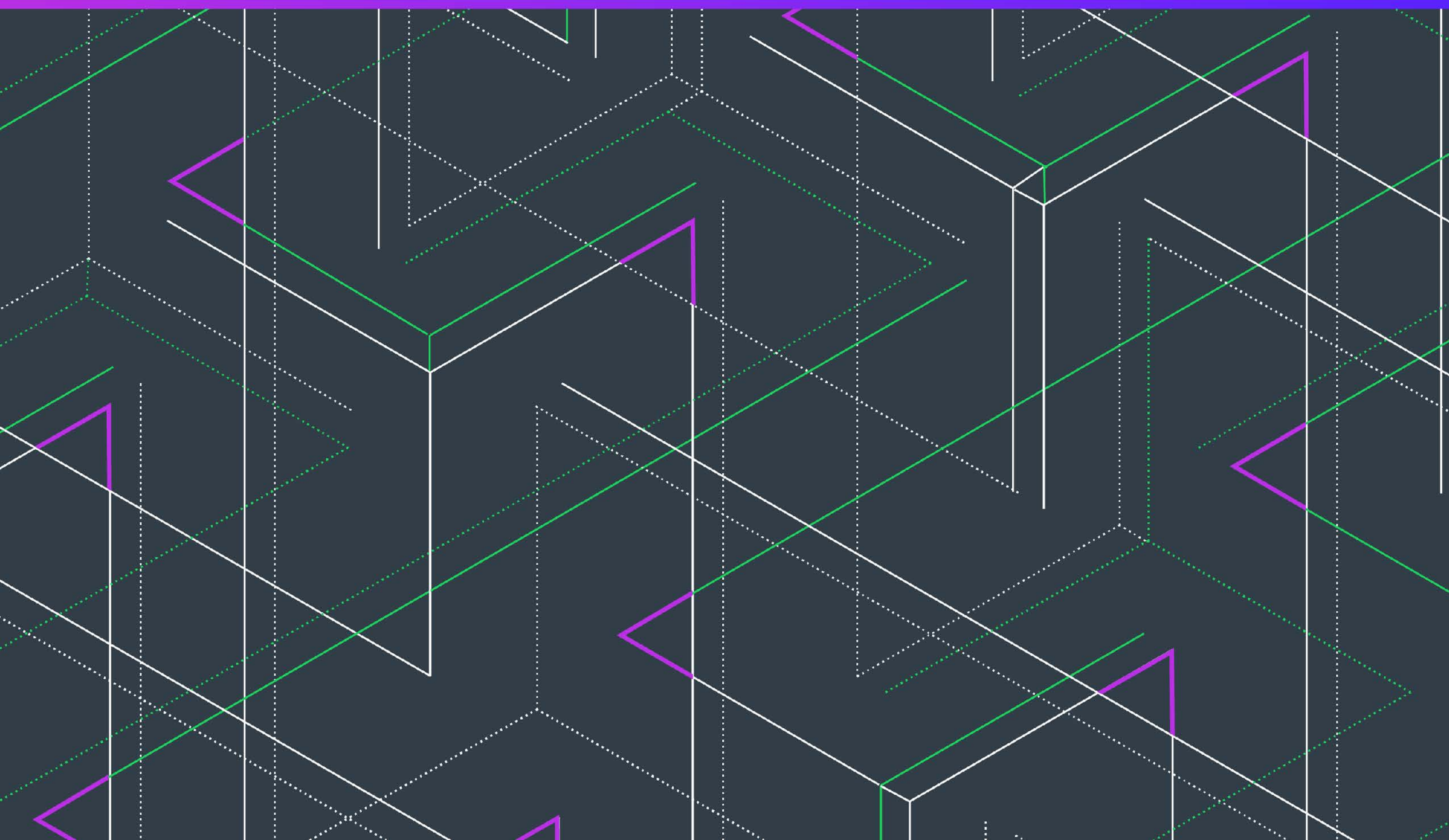
FlexNet Operations

Web Services Integration Guide



Legal Information

Book Name:	Flexnet Operations Web Services Integration Guide
Part Number:	FNO-2025R1-WSI00
Product Release Date:	October 2025

Copyright Notice

Intellectual Property

Restricted Rights Legend

Contents

1

FlexNet Operations Web Services Integration Guide

FlexNet Operations supports the integration of software license delivery into the business operations of software producers. Much of the FlexNet Operations functionality is exposed as SOAP-based Web Services to facilitate its integration with software producer back office systems. In addition, FlexNet Operations supports REST endpoints for managing entity-specific data extracts.

Some FlexNet Operations functionality is exposed for customization via web services. The protocol supported for implementing an external service depends on the functionality being customized.

Table 1-1 ▪ FlexNet Operations Web Services Integration Guide

Section	Description
Introduction	Introduces you to the types of web services supported by FlexNet Operations: ● SOAP-Based Web Service Operations ● RESTful Web Services ● External Web Services
Token-Based Authentication	Describes how to authenticate calls to SOAP and REST APIs using token-based authentication.
Using SOAP Web Services	Provides information about the SOAP services included in FlexNet Operations, where to find WSDLs for those services, and how to set up a Web Service client.
SOAP Web Services Reference	Link to complete reference information about FlexNet Operations SOAP-based APIs.
REST Web Services	Describes the REST services included with FlexNet Operations that producers can use to download data extract snapshots and manage their data extract job output as well as providing communication with end-user applications.

Table 1-1 ▪ FlexNet Operations Web Services Integration Guide

Section	Description
External Web Services	Describes how to create external services to customize aspects of FlexNet Operations behavior, and how to obtain sample code and other resources to facilitate these customization efforts.

SOAP-Based Web Service Operations

A client application can call FlexNet Operations Web Services to import product and entitlement data from a back-office system, generate licenses, and manage users and organizations. The chapter on SOAP-based services provides information about the SOAP services included in FlexNet Operations, where to find WSDLs for those services, and how to set up a Web Service client. It also discusses FlexNet Operations Web Service versions (which differ from the FlexNet Operations product version) and, for select services, include special usage tips.

- For more details about integrating your organization's systems with FlexNet Operations via the SOAP-based Web Service operations, see Using SOAP Web Services.
- For complete reference information about FlexNet Operations SOAP-based APIs, see the FlexNet Operations SOAP Web Services Guide.

***Tip ▪** To see the list of FlexNet Operations SOAP services, open a browser to `https://<host:port>/flexnet/services`, where `host` is your FlexNet Operations host and `port` is the port on which FlexNet Operations is running.*

RESTful Web Services

FlexNet Operations supports a number of REST Web Service endpoints that allow producers to extract snapshots of changes to fulfillments, devices, served devices, license models, usage activity, and more. These entity-specific extracts are generated by jobs that administrators can schedule in the Producer Portal. Then, producer users or applications developed by producers can use the REST endpoints to oversee and manage the data extracts generated by the jobs.

See REST Web Services for more information about how to use the data extract endpoints and the function and output of the data extract jobs.

***Tip ▪** The data extract jobs do not run by default. Producers can configure one or more of the data extract jobs to run in the Producer Portal (**Administer** > **Configure Alerts/Jobs**). For more information about managing data extract jobs and other alerts, see Administering Alerts in the FlexNet Operations User Guide.*

External Web Services

Producer can extend or customize the way some features of FlexNet Operations function by implementing external services on their own servers. Most external services can be implemented with SOAP services but some, like the capability request callout, support REST services.

See External Web Services for more information about functionality that can be customized and guidance about how to implement an external service.

Product Support Resources

The following resources are available to assist you with using this product:

- Revenera Product Documentation
- Revenera Community
- Revenera Learning Center
- Revenera Support

Revenera Product Documentation

You can find documentation for all Revenera products on the Revenera Product Documentation site:

https://docs.revenera.com

Revenera Community

On the Revenera Community site, you can quickly find answers to your questions by searching content from other customers, product experts, and thought leaders. You can also post questions on discussion forums for experts to answer. For each of Revenera's product solutions, you can access forums, blog posts, and knowledge base articles.

https://community.revenera.com

Revenera Learning Center

The Revenera Learning Center offers free, self-guided, online videos to help you quickly get the most out of your Revenera products. You can find a complete list of these training videos in the Learning Center.

https://learning.revenera.com

Revenera Support

For customers who have purchased a maintenance contract for their product(s), you can submit a support case or check the status of an existing case by first logging into the Revenera Community, clicking **Support** on the navigation menu to open the **Support Hub** page, and then clicking the **Open New Case** or **Case Portal** button.

Contact Us

Revenera is headquartered in Itasca, Illinois, and has offices worldwide. To contact us or to learn more about our products, visit our website at:

http://www.revenera.com

You can also follow us on social media:

- Twitter

- Facebook
- LinkedIn
- YouTube
- Instagram

2

Token-Based Authentication

FlexNet Operations users can use a token-based authentication mechanism to securely authorize SOAP and REST web service calls to FlexNet Operations.

In token-based authentication, the client first authenticates with their credentials. If successful, they receive an access token, which then must be included as a Bearer token in the Authorization HTTP header for subsequent requests. The access token has a limited duration and contains authorization information, but no identity information.

FlexNet Operations users manage bearer tokens using the REST web service **access-token-controller**. This REST web service supports creating, reading, updating, searching, rotating, and deleting tokens, as well as using the tokens to call the APIs.

Note* ▪** *Tokens can also be managed using the* ***Manage Access Tokens *page in the Producer Portal or End-User Portal. For more information, see the section Managing Access Tokens in the FlexNet Operations User Guide or Managing Access Tokens in the FlexNet Operations End-User Portal Help. Both documents are available from the Revenera Documentation site.*

This section covers the following:

- Which APIs Support Token-Based Authentication?
- Key Information for the access-token-controller REST Web Service
- Token Types and Their Uses
- Token Attributes
- Managing Tokens

Which APIs Support Token-Based Authentication?

The following APIs support token-based authentication:

- SOAP web services (as listed in Summary of SOAP Web Service Operations)
- Data Extract RESTful Services
- Data Extract RESTful ServicesData Extract RESTful Services
- Application RESTful Services

***Note** ▪ At present, token-based authentication is not available for these APIs:*

- *Data Extract RESTful Services*
- *Event Notifications*

Key Information for the access-token-controller REST Web Service

This section covers the following aspects:

- Workflow
- Permissions for access-token-controller
- Base URL for access-token-controller
- Additional Documentation for access-token-controller

Workflow

FlexNet Operations users call the **access-token-controller** REST web service to request an access token with a limited expiration from FlexNet Operations. The web service returns an access token which is a hexadecimal string with an `rna_` prefix.

When firing requests to a FlexNet Operations API (REST or SOAP), this token must be added to the authentication header of the API request with the word `Bearer` followed by the token string.

The access token not only authenticates the requester but also defines the permissions of how the requester can use the API.

Permissions for access-token-controller

Any FlexNet Operations user can create and manage NORMAL tokens for themselves.

To create and manage tokens for other users (IMPERSONATED tokens), users require the following permissions:

- **Create Impersonated Token**
- Plus at least one of the following permissions, depending on the type of user creating the IMPERSONATED token:
    - **View and Manage Users**
    - **View and Manage Customer Users**
    - **View and Manage Partner Users**

Base URL for access-token-controller

Endpoints for the **access-token-controller** REST API are available on your FlexNet Operations host:

```
https://<siteID>.flexnetoperations.com/uar/v1/tokens
```

```
https://<siteID>.flexnetoperations.com/uar/v1/token
```

where <siteID> is your organization’s site ID which is supplied by Revenera. This is usually your organization's DNS name, but can also be the tenant ID that was assigned to your organization when FlexNet Operations was first implemented, and would have the format **flex*NNNN***.

Note ▪ *URLs in this document refer to v1 of the APIs. Further releases of the API may increment this version. Please consult the Swagger documentation to identify the latest version of the API, and modify the URLs you use appropriately.*

Additional Documentation for access-token-controller

In addition to this manual, the following documentation resources are available:

Redocly

Use the following URL to access the Redocly documentation:

https://fnoapi-access-token-controller.redoc.ly/

The **access-token-controller** pages on the Redocly platform provide the same information as the **access-token-controller** pages on Swagger, but in a slightly different format.

Swagger

Use the following URL to access the **access-token-controller** REST API documentation on Swagger:

```
https://<siteID>.flexnetoperations.com/flexnet/swagger-ui.html#/access45token45controller
```

where <siteID> is your organization’s site ID which is supplied by Revenera. This is usually your organization's DNS name, but can also be the tenant ID that was assigned to your organization when FlexNet Operations was first implemented, and would have the format **flex*NNNN***.

Token Types and Their Uses

FlexNet Operations uses different token types to manage different access levels and operations:

- NORMAL tokens can be requested by any FlexNet Operations user to authenticate themselves at a FlexNet Operations API.
- IMPERSONATED tokens can be requested by a privileged user for use by other FlexNet Operations users.

  An IMPERSONATED token would typically be requested by a privileged user on behalf of another user with limited permissions. This enables the privileged user to access the web services securely and to act as the other user temporarily, with their actions logged.

  In addition, in scenarios where a custom portal was built using FlexNet Operations APIs, a privileged user impersonating another user will see the content that is specific to the user that they are impersonating.

***Important** ▪ To create and manage IMPERSONATED tokens, users require the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller).*

Token Attributes

A request to the **access-token-controller** REST web service must specify the following attributes for each token.

Table 2-1 ▪ Mandatory token attributes

Attribute	Data Type	Description
expiryStr	string	Lifetime of the token, with a minimum lifetime of 1 minute. Use the following format to specify the token lifetime: Y=years, M=months, d=days, h=hours, m=minutes (case sensitive). **Example**: `3Y 4M 3d 9h 6m` Best practice: Use a short duration and rotate tokens once every month or three months. See also Rotating a Token.
tokenName	string	Custom token name specified by the user. The name must be 5 - 25 characters long. The following characters are not supported: `* < > + $ ? . ^ | % ] \\\\` (four consecutive back slashes are not allowed) HTML tags that may indicate an XSS attack will be rejected.
tokenDescription	string	A description of the token. This field is mandatory for IMPERSONATED tokens and should capture the impersonation reason.
tokenType	string	The token type. Valid values: NORMAL, IMPERSONATED. See also Token Types and Their Uses.
username	string	The user for whom the token is created. This field is mandatory for IMPERSONATED tokens.

In addition to the attributes above, the response to a token request will include the following attributes.

Table 2-2 ▪ Token attributes included in the response to a request for token

Attribute	Data Type	Description
tokenExpiryMillis	integer	The date and time when the token expires in milliseconds since 1970-01-01 00:00.
tokenIssueMillis	integer	The date and time when the token was issued in milliseconds since 1970-01-01 00:00.

Table 2-2 ▪ Token attributes included in the response to a request for token

Attribute	Data Type	Description
tokenCreator	string	The ID of the logged-in user who is requesting the token. This is usually an email address.

Token Creator vs User Name

Note the difference between the token creator and the user name:

- For NORMAL tokens, the value for `tokenCreator` will be identical with the value for `username`.
- For IMPERSONATED tokens:
  - The `tokenCreator` is the logged-in user requesting the token.
  - The `username` specifies the user for whom the token is generated.

Managing Tokens

This section covers the following token operations:

- Creating a Token
- Verifying a Token
- Rotating a Token
- Deleting a Token
- Getting Token Details
- Updating a Token
- Listing Non-Expired Tokens
- Getting the Count of Non-Expired Tokens
- Searching for Tokens

Important* ▪** *To perform any of the above mentioned tasks for IMPERSONATED tokens, users require the* ***Create Impersonated Token *permission and a* ***Manage Users*** *permission (see Permissions for access-token-controller).*

Creating a Token

Use the **/token** endpoint to create a token.

For `tokenType`, specify either NORMAL or IMPERSONATED. For more information on these values, see Token Types and Their Uses. For information about the remaining attributes in the request body, see Token Attributes.

Important ▪ *To create and manage IMPERSONATED tokens, users require the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller).*

Item	Description
URI	`/uar/v1/token`
Method	POST
Query parameters	N/A
Request body	`{ "expiryStr": "string", "tokenDescription": "string", "tokenName": "string", "tokenType": "NORMAL | IMPERSONATED", "username": "string" }`
Response codes	201: The token was successfully created.

Sample Request

The following request creates an IMPERSONATED token that is valid for 10 minutes and associated with the user name **fnouser@mycompany.com**.

```
{
    "expiryStr": "10m",
    "tokenDescription": "Impersonating fnouser",
    "tokenName": "demo",
    "tokenType": "IMPERSONATED",
    "username": "fnouser@mycompany.com"
}
```

Sample Response

The following shows a sample response for the successful creation of an IMPERSONATED token.

```
{
    "statusMessage": "Successful",
    "responseObject": {
        "expiryStr": "10m",
        "tokenExpiryMillis": 1716900570355,
        "tokenIssueMillis": 1716899970355,
        "tokenName": "demo",
        "tokenCreator": "systemadmin@mycompany.com",
        "tokenType": "IMPERSONATED",
        "tokenValue": "rna_5dcd6233abs227140394653da3a328ad5556b058d87",
        "username": "fnouser@mycompany.com"
    },
}
```

The token creator—`systemadmin@mycompany.com`—is the logged-in user who requested the token. They are impersonating the user with the ID `fnouser@mycompany.com`, to access the API on their behalf.

Make a note of the token value—in this example, `rna_5dcd6233abs227140394653da3a328ad5556b058d87`. This is the only time the token value is displayed. This value must be included as a Bearer token in the Authorization HTTP header for authorization requests.

Verifying a Token

Calling the **/token/verification** endpoint allows you to verify if a token is valid. In the request body, pass the token value—a 44-character long hexadecimal string with an `rna_` prefix—that was generated when calling **/token**.

Item	Description
URI	`/uar/v1/token/verification`
Method	`POST`
Query parameters	N/A
Request body	`{ "accessToken": "string" }`
Response codes	`200`: Success

Sample Request

```
    {
        "accessToken": "rna_5dcd6233abs227140394653da3a328ad5556b058d87"
    }
```

Sample Response

The following shows a sample response with details of the token (response code `200`, `Success`):

```
{
    "statusMessage": "Successful",
    "responseObject": {
        "expiryStr": "10m",
        "tokenExpiryMillis": 1716900570355,
        "tokenIssueMillis": 1716899970355,
        "tokenName": "demo",
        "tokenCreator": "systemadmin@mycompany.com",
        "tokenType": "IMPERSONATED",
        "username": "fnouser@mycompany.com"
    },
}
```

Rotating a Token

Token rotation is a security best practice that involves regularly replacing old tokens with new ones in a system where tokens are used for authentication and authorization. This practice enhances security and maintains the integrity of access control mechanisms. Token rotation also enables organizations to quickly generate a new token if the existing token value is compromised. Revenera recommend to frequently rotate access tokens (the frequency will depend on your organization's security policies).

Token rotation means that a token's properties (expiry strategy, expiry time, issue time, type, name and user name) are retained, but a new token value is created. No values can be changed.

If a token is rotated halfway through its lifetime (specified in `expiryStr`), the token expiry time (`tokenExpiryMillis`) does not change. For example, if a token is created on January 1 with a lifetime of 30 days (`"expiryStr": "30d"`), and the token is rotated on January 20, its expiry time of January 30 (`"tokenExpiryMillis": 1704067200000`) remains unchanged.

Call the **/token/{tokenName}/rotation** endpoint using `POST`, where you pass the token name as a path parameter.

Item	Description
URI	`/uar/v1/token/{tokenName}/rotation`
Method	`POST`
Query parameters	N/A
Request body	N/A
Response codes	`200`: Success

Sample Response

The following shows a sample response with details of the rotated token (response code `200`, `Success`).

It is important that you make a note of the rotated token value; this is the only time the new token value is displayed.

```
{
    "statusMessage": "Successful"
    "responseObject": {
        "expiryStr": "10m",
        "tokenDescription": "demo",
        "tokenExpiryMillis": 1716900570355,
        "tokenIssueMillis": 1716899970355,
        "tokenName": "demo",
        "tokenCreator": "systemadmin@mycompany.com",
        "tokenType": "NORMAL",
        "tokenValue": "rna_164dabe428eff69835a25ba555d9c22714033abs227",
        "username": "systemadmin@mycompany.com"
    },
}
```

Deleting a Token

You can delete a token in case they become compromised.

Only the token creator can delete a token; this means that only users with the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller) can delete IMPERSONATED tokens.

Once a token has been deleted it can no longer be used to access the APIs.

Call the **/token/{tokenName}** endpoint using `DELETE`, where you pass the name of the token to be deleted as a path parameter. Note that the token name is case sensitive.

Item	Description
URI	`/uar/v1/token/{tokenName}`
Method	`DELETE`
Query parameters	N/A
Request body	N/A
Response codes	`204`: Success, no content.

Sample Response

For a successful DELETE, the status code 204 is returned, but no response is provided.

Getting Token Details

You can retrieve the details for a specified token. Call the **/token/{tokenName}** endpoint using `GET`, where you pass the token name as a path parameter. Note that the token name is case sensitive.

Item	Description
URI	`/uar/v1/token/{tokenName}`
Method	`GET`
Query parameters	N/A
Request body	N/A
Response codes	`200`: Success

Sample Response

The following shows a sample response with details of the token (response code `200`, `Success`):

```
{
    "statusMessage": "Successful",
    "responseObject": {
        "expiryStr": "10m",
        "tokenExpiryMillis": 1716900570355,
        "tokenIssueMillis": 1716899970355,
        "tokenName": "demo",
        "tokenCreator": "systemadmin@mycompany.com",
        "tokenType": "NORMAL",
        "username": "systemadmin@mycompany.com"
    },
}
```

Updating a Token

Only the token creator or another privileged user can update a token. The following attributes of an existing token can be edited:

- `tokenName`
- `tokenDescription`
- `expiryStr`

***Note** ▪ You cannot change the token expiry to a value in the past.*

You cannot change the user name or token type. Expired tokens cannot be updated.

***Note** ▪ For a description of token attributes, see Token Attributes.*

Call the **/token/{tokenName}** endpoint using PUT, where you pass the name of the token that you want to update as a path parameter. Note that the token name is case sensitive.

In the request body, pass the parameters that you want to update.

Item	Description
URI	`/uar/v1/token/{tokenName}`
Method	PUT
Query parameters	N/A
Request body	`{ "expiryStr": "string", "tokenDescription": "string", "tokenName": "string", }`
Response codes	204: Success, no content.

Sample Request

The following request updates the expiry, token description and token name.

```
{
    "expiryStr": "10M",
    "tokenDescription": "updateddescription",
    "tokenName": "updatedtoken",
}
```

Sample Response

If the request is successful, the status code 204 is returned, but no response is provided.

Listing Non-Expired Tokens

Call the **/tokens** endpoint using `GET` to search for non-expired tokens for a specified user. You can search for either NORMAL or IMPERSONATED tokens for the user:

- NORMAL tokens—Specify the name or ID (email address) of the user *for whom tokens were generated* (as specified in the `username` field when POSTing to **/uar/v1/token**) in the `username` parameter.
- IMPERSONATED tokens—Specify the name or ID (email address) of the user *who requested a token on behalf of another user* (as returned in the `tokenCreator` field in the response to a token creation request) in the `tokenCreator` parameter.

  Important ▪ *You need the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller) to get a list of IMPERSONATED tokens.*

You must specify a value for either the `username` or `tokenCreator` parameter. If you pass values for both parameters in the same request, the API returns a list of all non-expired tokens that match both parameters.

Important ▪ *If you do not have permission to view a given token, its details may be returned as masked entries.*

Tip ▪ *If you want a more fine-grained search, use **/tokens/search**. Use the **/tokens/count** API to determine the number of tokens for a specified user.*

Item	Description
URI	`/uar/v1/tokens`
Method	GET

Item	Description
Query parameters	● username (case sensitive)—Pass the name of the user for whom tokens were generated to get a list of all NORMAL tokens. If you do not pass the username parameter, you must pass the tokenCreator parameter. ● tokenCreator (case sensitive)—Pass the name of the user who requested tokens on behalf of another user to get a list of all IMPERSONATED tokens. If you do not pass the tokenCreator parameter, you must pass the username parameter. ● page (optional)—Retrieves a specific page. Must be a number (starting from 0). ● pagesize (optional)—Specifies how many tokens per page should be returned. Must be a number.
Request body	N/A
Response codes	200: Success 400: Unless you have the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller), you cannot access tokens for other users.

Sample Request

The following shows an example for a Curl call to get unexpired IMPERSONATED tokens for user **admin@mycompany.com**, with pagination options 1 page and a maximum of 2 records per page:

```
curl -X GET --header 'Accept: application/json' 'https://<siteID>.flexnetoperations.com/flexnet/uar/v1/
tokens?tokenCreator=admin%40mycompany.com&page=1&pagesize=2'
```

Sample Response

The following shows a sample response listing the IMPERSONATED access tokens for user **admin@mycompany.com**:

```
{
    "statusMessage": "Successful",
    "responseObject": [
        {
            "expiryStr": "0y 2M 25d 2h 50m",
            "tokenName": "demo2",
            "tokenDescription": "demo2-description",
            "tokenType": "IMPERSONATED",
            "username": "fnouser2@mycompany.com",
            "tokenIssueMillis": 1721308129531,
            "tokenExpiryMillis": 1731762710070,
            "tokenCreator": "admin@mycompany.com"
        },
        {
            "expiryStr": "10M",
            "tokenName": "demo3",
            "tokenDescription": "demo3-description",
            "tokenType": "NORMAL",
            "username": "fnouser3@mycompany.com",
            "tokenIssueMillis": 1724329674453,
            "tokenExpiryMillis": 1750596280485,
            "tokenCreator": "admin@mycompany.com"
        }
    ]
}
```

Getting the Count of Non-Expired Tokens

Call the **/tokens/count** endpoint using `GET` to return a list of non-expired tokens for a specified user. You can get a count of NORMAL or IMPERSONATED tokens for the user:

- NORMAL tokens—Specify the name or ID (email address) of the user *for whom tokens were generated* (as specified in the `username` field when POSTing to **/uar/v1/token**) in the `username` parameter.
- IMPERSONATED tokens—Specify the name or ID (email address) of the user *who requested a token on behalf of another user* (as returned in the `tokenCreator` field in the response to a token creation request) in the `tokenCreator` parameter.

Important *▪ You need the* ***Create Impersonated Token*** *permission and a* ***Manage Users*** *permission (see Permissions for access-token-controller) to get the count of IMPERSONATED tokens.*

Tip *▪ Use the* ***/tokens*** *API to get a list of non-expired tokens for a specified user.*

Item	Description
URI	`/uar/v1/tokens/count`
Method	`GET`
Query parameters	● `username` (case sensitive)—Pass the name of the user for whom tokens were generated to get a list of all NORMAL tokens. If you do not pass the `username` parameter, you must pass the `tokenCreator` parameter. ● `tokenCreator` (case sensitive)—Pass the name of the user who requested tokens on behalf of another user to get a list of all IMPERSONATED tokens. If you do not pass the `tokenCreator` parameter, you must pass the `username` parameter.
Request body	N/A

Item	Description
Response codes	`200`: Success `400`: Unless you have the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller), you cannot access tokens for other users.

Sample Request

The following shows an example for a Curl call to get the number of unexpired NORMAL tokens for user **fnouser3@mycompany.com**.

```
curl -X GET --header 'Accept: application/json' 'https://<siteID>.flexnetoperations.com/flexnet/uar/v1/
tokens/count?username=fnouser3%40mycompany.com'
```

Sample Response

The following shows a sample response for the count of unexpired NORMAL tokens returned for user **fnouser3@mycompany.com**:

```
{
    "statusMessage": "Successful",
    "responseObject": 2
}
```

Searching for Tokens

Call the **/tokens/search** endpoint using POST to return a list of non-expired tokens that match the search criteria that you specify. This API is similar to **/tokens**, but enables a much more fine-grained search.

Note ▪ *If you do not have permission to view a given token, its details may be returned as masked entries.*

Tip ▪ *Use the **/tokens** API to get a list of non-expired tokens for a specified user.*

Item	Description
URI	`/uar/v1/tokens/search`
Method	POST

Item	Description
Query parameters	For descriptions, see Search Parameters, below. ● `tokenCreator` (string, case sensitive) ● `expiresBefore` (string, case sensitive) ● `expiresLaterThan` (string, case sensitive) ● `issuedBefore` (string, case sensitive) ● `page` (integer, required) ● `pageSize` (integer, required) ● `tokenName` (string, case sensitive) ● `tokenType` (string, case sensitive) ● `username` (string, case sensitive)

Request body	`{`
	`    "tokenCreator": "string",`
	`    "expiresBefore": "string",`
	`    "expiresLaterThan": "string",`
	`    "issuedBefore": "string",`
	`    "page": 0,`
	`    "pageSize": 0,`
	`    "tokenName": "string",`
	`    "tokenType": "NORMAL | IMPERSONATED",`
	`    "username": "string"`
	`}`

Response codes	`200`: Success `400`: Unless you have the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller), you cannot access tokens for other users.

Search Parameters

In the request body, specify at least one of the following search criteria: `tokenCreator`, `username`, `expiresBefore`, `expiresLaterThan`, `issuedBefore`, `tokenName`.

If you pass values for the `username` and `tokenCreator` parameters in the same request, the API returns a list of all non-expired tokens that match both parameters.

Table 2-3 ▪ Search parameters for returning non-expired tokens

Parameter	Description	Notes
tokenCreator	The user name (email address) of the user who created the token. Use an asterisk (*) as placeholder for partial searches. ***Important ▪** You need the **Create Impersonated Token** permission and a **Manage Users** permission (see Permissions for access-token-controller) to search for IMPERSONATED tokens.*	For NORMAL tokens, `tokenCreator` and `username` (if specifying both) must be the same user.
username	The user name (email address) of the user on whose behalf the token was created (in other words, the user who is being impersonated).	
expiresBefore	Specify to return tokens that expire before a set interval. If specified, must be greater than the `expiresLaterThan` parameter.	Use the following format to specify the interval: `Y`=years, `M`=months, `d`=days, `h`=hours, `m`=minutes (case sensitive). For example: `3Y 4M 3d 9h 6m`
expiresLaterThan	Specify to return tokens that expire after a set interval. If specified, must be less than the `expiresBefore` parameter.	
issuedBefore	Specify to return tokens that were issued before a set interval.	
tokenName	Specify the name of the token.	You can use the wildcard character * when you need a partial match, or if you are unsure of the exact token name.
tokenType	Specify NORMAL or IMPERSONATED. If not specified, tokens of both types will be returned.	N/A
page	Mandatory. Specify the number of the page that should be returned. Values are zero based.	N/A
pageSize	Mandatory. Specify how many tokens should be returned per page.	N/A

Sample Request

The following shows an example request to search for unexpired NORMAL tokens called "demo*" for user **fnouser3@mycompany.com** that expire within the next 3 months:

```
{
    "tokenCreator": "fnouser3@mycompany.com",
    "expiresBefore": "1y",
    "expiresLaterThan": "1M",
    "issuedBefore": "",
    "page": 0,
    "pageSize": 2,
    "tokenName": "demo*",
    "tokenType": "NORMAL",
    "username": ""
}
```

This is the equivalent Curl call:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
    "tokenCreator": "fnouser3@mycompany.com", \
    "expiresBefore": "3M", \
    "expiresLaterThan": "", \
    "issuedBefore": "", \
    "page": 0, \
    "pageSize": 2, \
    "tokenName": "demo*", \
    "tokenType": "NORMAL", \
    "username": "" \
 }' 'https://<siteID>.flexnetoperations.com/flexnet/uar/v1/tokens/search'
```

Sample Response

The following shows a sample response:

```
{
    "statusMessage": "Successful",
    "responseObject": {
        "response": [
            {
              "expiryStr": "0y 3M",
              "tokenName": "demo1",
              "tokenDescription": "demo token 1",
              "tokenType": "NORMAL",
              "username": "fnouser3@mycompany.com",
              "tokenIssueMillis": 1724347771843,
              "tokenExpiryMillis": 1732300171843,
              "tokenCreator": "fnouser3@mycompany.com"
            },
            {
              "expiryStr": "0y 2M 25d 2h 50m",
              "tokenName": "demo2",
              "tokenDescription": "demo token 2",
              "tokenType": "NORMAL",
              "username": "fnouser3@mycompany.com",
              "tokenIssueMillis": 1721308129531,
              "tokenExpiryMillis": 1731762710070,
              "tokenCreator": "fnouser3@mycompany.com"
```

```
                }
            ],
            "totalResults": 3,
            "pageNumber": 0,
            "pagesize": 2
        }
    }
```

3

Using SOAP Web Services

FlexNet Operations SOAP Web Services are accessed using a document-style SOAP mechanism from client applications that you write. SOAP is an XML-based protocol that supports information exchange using HTTP and has implementations in many programming languages. Because each Web Service is a document-style SOAP service, its parameters are sent in a specific XML format. This format is defined in a service-specific XML Schema document and a WSDL file. A language-specific client implementation (for example, C, C++, Java, Visual Studio .NET, Perl) of each interface can be generated from its WSDL file.

A WSDL file defines the interface to each Web Service. The custom data types and XML Schema elements specific to each interface are defined in the *FlexNet Operations SOAP Web Services Guide*. The files containing these interface definitions are located at `https://<host:port>/flexnet/services`, where *host* and *port* specify the location of the running FlexNet Operations application.

Summary of SOAP Web Service Operations

A client application can call FlexNet Operations Web Services to import product and entitlement data from a back-office system, generate licenses, and manage users and accounts. FlexNet Operations supports the following Web Services, each providing a number of operations.

Table 3-1 ▪ Summary of FlexNet Operations Web Services

SOAP Web Service	Description
Product Packaging Service	● Create, update, and delete features, feature bundles, products, suites, and maintenance. ● Set the state of features, feature bundles, products, suites, and maintenance. ● Get the number of features, feature bundles, products, suites, and maintenance and the entities themselves that match specified criteria. ● Create, update, and delete relationships between products, suites, and maintenance. ● Create and delete unassigned part numbers. ● Get the identifiers for existing license models. ● Get the identifiers for existing transaction keys. ● Get license technologies and their license generator configurations. ● Manage product lines.
Entitlement Order Service	● Get entitlement-time license model attributes. ● Create, update, and delete simple and bulk entitlements and their line items. ● Set the state of an entitlement. ● Email entitlement certificates to customers. ● Get the number of existing entitlements and activatable line items and the entitlements and activatable line items themselves that match specified criteria. ● Link and unlink maintenance line items and their parent/associated line items. ● Merge entitlements. ● Transfer entitlements and line items. ● Split line items and bulk entitlements, and get matching line items and bulk entitlements to receive the splits. ● Renew, upgrade, and upsell entitlements. ● Import web register keys; get the number of web register keys and the web register keys themselves that match specified criteria. ● Map activation and entitlement IDs to self-registered End-User Portal users.

Table 3-1 ▪ Summary of FlexNet Operations Web Services (cont.)

SOAP Web Service	Description
License Service	● Get fulfillment-time license model attributes. ● Verify the generation of certificate licenses. ● Generate certificate licenses, individually or in batch mode. ● Email, return, rehost, and repair certificate licenses. ● Consolidate licenses and email consolidated licenses. ● Activate, return, and repair licenses with short codes. ● Get existing simple and consolidated fulfillments that match specified criteria. ● Get host attributes from a license technology. ● Set the license text for on-hold fulfillments. ● Delete on-hold fulfillments. ● Submit an offline activation request for trusted activation. ● Transfer entitlements and fulfillments on a particular host from one customer account to another.
User Account Hierarchy Service	● Add, update and delete account hierarchy information. ● Link accounts in hierarchies. ● Relate partner accounts. ● Add, update and delete user information. ● Update user roles. ● Query for parent accounts, subaccounts, and related partner accounts.
FlexNet Authentication Service	● Provide a token to an external application to support a single sign-on mechanism for the external application and FlexNet Operations.
Device Management Service	● Create, update, and delete client, server, and test devices. ● Set the state of devices. ● Get the number of devices and the devices themselves that match specified criteria. ● Generate pre-installed licenses for devices. ● Generate a capability response from a submitted capability request or a capability request synthesized from submitted input data. ● Link and remove product line items.

Table 3-1 ▪ Summary of FlexNet Operations Web Services (cont.)

SOAP Web Service	Description
Download Packaging Service	● Add-on to FlexNet Operations Entitlement Management Service. ● Components are files and download packages. ● Includes additional attributes of End User License agreements.
Usage Service	● Retrieve the end-user’s product line, meter, unit of measure, actual use, entitled quantity, overage since statement, overage since reset, percentage, interval/reset, and time period.

To see the list of FlexNet Operations services, open a browser to `https://<host:port>/flexnet/services`.

Note that the URL for the Download Packaging Service differs from the URL above. The Download Packaging Service is available at `https://<siteID>-fno.flexnetoperations.com/flexnet/services/<version>/DownloadPackagingService`, where <siteID> is your organization’s site ID which is supplied by Revenera.

For a description of Download Packaging Service versions, refer to Web Service Version History.

Important ▪ *Several other Web Services are displayed on this page. The `AdminService` and `Version` Web Services are specific to Axis. The `ActivationService` is a proprietary Web Service. Only the Web Service operations documented in this manual are intended to be invoked directly by a producer’s client application.*

SOAP Web Service Versioning

FlexNet Operations Web Services use a separate versioning system from the overall FlexNet Operations product. To provide producers with greater flexibility when managing their own Web Service client code, new Web Service versions are created as new features are added to existing services.

The base version of each Web Service that exists in the root of the Web Services directory is the original versions introduced in FlexNet Operations 2016 and, in some cases, modified in FlexNet Operations 2016 Release 2. New versions are indicated by a version designator: V1, V2, and so forth. Web Services with no additional versions listed offer only the base version. Producers can choose to use the newer version of a given Web Service or to continue using a prior release’s Web Service version in their Web Service clients.

To see a listing of FlexNet Operations Web Services, open a Web browser and navigate to

`https://<host:port>/flexnet/services/`

where *`host`* is your FlexNet Operations host and *`port`* is the port on which FlexNet Operations is running.

Working with Web Service Versions

As each Web Service changes in a way that could pose a risk for backwards compatibility, FlexNet Operations creates a new version of that Web Service. The base version of each service remains intact to preserve functionality of all existing Web Service clients.

To take advantage of a new Web Service version, producers must update their Web Service client code to reference the new version's WSDL and endpoint URL, and then make any additional code alterations necessary to take advantage of the new version's functionality.

For example, to use the v1 of the Product Packaging service, reference the WSDL at:

```
https://<host:port>/flexnet/services/v1/ProductPackagingService?wsdl
```

where `*host*` is your FlexNet Operations host and `*port*` is the port on which FlexNet Operations is running. Then make any additional code changes necessary to adapt your Web Service client to use the new Web Service version.

Web Service Version History

The following table identifies the available versions for each Web Service and identifies the release for which the version was introduced.

Table 3-2 ▪ FlexNet Operations Web Services Versions

Web Service	Versions	Changes
Product Packaging	**v3**	Introduced in FlexNet Operations 2024.06, v3 changes include: ● The **getLicenseModelIdentifier** operation now returns the following license model details: ● Expiration (duration) ● Embedded or Not Embedded ● Counted or Uncounted ● Grace Period.
	v2	Introduced in FlexNet Operations 2018 R1, v2 changes include: ● Added optional parameter **productLine** for the **getProductsQuery** operation to return details of all products that are linked with a particular product line. ● Changing **organization** to **account** and **org** to **acct** ● Removed the deprecated **createAccount** (formerly **createOrganization**) method. ● Added optional parameter **AggregationType** for the **createFeatureRequest** operation to specify a feature count aggregation type. Valid values are `SUM`, `MAX`, and `NONE`.
	v1	Introduced in FlexNet Operations 2016 Release 3, v1 changes include ● New attributes to products to support the specification of upgrade start and end dates and the email template variation to use for product version upgrade notifications. ● Web Service versioning introduced.
	base	The original Product Packaging service introduced in FlexNet Operations 2016.
Entitlement Order	**v8**	Introduced in FlexNet Operations 2024.11, v8 changes include: ● Edited **getEntitlementLineItemPropertiesQuery** to allow searching by multiple account names. The `<urn:value>` element of the complex element `<urn:accountUnitName>` can be repeated multiple times to search by multiple account names.

Table 3-2 ▪ FlexNet Operations Web Services Versions (cont.)

Web Service	Versions	Changes
Entitlement Order (continued)	**v7**	Introduced in FlexNet Operations 2024.05, v7 changes include: ● Added the properties **isAutoRenewal**, **renewalInterval** and **renewalExpirationDate** to enable automatic renewal of entitlement line items. These properties are available for the following operations: ● searchEntitlementDataType ● createEntitlementLineItemDataType ● updateLineItemDataType ● searchActivatableItemDataType ● entitlementLineItemResponseConfigRequestType ● entitlementLineItemPropertiesType ● **updateEntitlementLineItem** has an additional element **lineItemType** under the **parentLineItem** element, to enable adding/removing Upgrade relationship between line items.
	v6	Introduced in FlexNet Operations 2023.08, v6 changes include: ● Added parameter tag **<urnactivationIds> ... </urn:activationIds>** to the **getActivatableItemsQuery** operation, to enable mass entitlement search. ● Added optional parameter **BooleanValue** to the **getEntitlementsQuery** method to enable search for entitlements that have a particular Boolean custom attribute applied.
	v5	Introduced in FlexNet Operations 2021.09, v5 changes include: ● Added optional parameter **lineItemState** to the **getEntitlementLineItemPropertiesQuery** operation. ● Binding name format omits the "/". ● Added optional parameter **isEmbeddedLicenseModel** to the **getEntitlementLineItemPropertiesQuery** operation.
	v4	Introduced in FlexNet Operations 2018 R3.
	v3	Introduced in FlexNet Operations 2018 R1, v3 changes include: ● Changing **organization** to **account** and **org** to **acct**
	v2	Introduced in FlexNet Operations 2017 R4, v2 changes include: ● Added **transferredFromLineItem** and **splitFromLineItem** to the response for the **getEntitlementLineItemPropertiesQuery**

Table 3-2 ▪ FlexNet Operations Web Services Versions (cont.)

Web Service	Versions	Changes
Entitlement Order (continued)	**v1**	Introduced in FlexNet Operations 2016 R4, v1 changes include ● Removal of **entitledProducts** from **updateBulkEntitlement**. ● Web Service versioning introduced.
	base	The original Entitlement Order service introduced in FlexNet Operations 2016.
License	**v2**	Introduced in FlexNet Operations 2025.02, v2 changes include: ● Added **getHostQuery** operation to list fulfillment and license information by hostid. ● Added **getHostCount** operation to get the batch size that can be used as input in **getHostQuery**.
	v1	Introduced in FlexNet Operations 2018 R1, v1 changes include: ● Changing **organization** to **account** and **org** to **acct**
	base	The original License service introduced in FlexNet Operations 2016.
User Account Hierarchy	**v6**	Introduced in FlexNet Operations 2024.07, v6 changes include: ● **accountID** parameter has been restructured as **accountIDList**. **accountIDList** supports the parameter **accountID** inside **accountIds** to support retrieving information for up to 10 accounts in the **getUsersQuery** method.
	v5	Introduced in FlexNet Operations 2023.06, v5 changes include: ● Added **name** field to the **acctRolesList** response in the **getUsersQuery** method.
	v4	Introduced in FlexNet Operations 2021.09, v4 changes include: ● Binding name format omits the “/”.
	v3	Introduced in FlexNet Operations 2018 R3, v3 changes include: ● Added **createdDate**, **createdBy** and **lastModifiedBy** fields to the **getAccountsQuery**. ● Added **createdBy** and **lastModifiedBy** fields to the **getUsersQuery**
	v2	Introduced in FlexNet Operations 2018 R1, v2 changes include: ● Rename user related fields from **userAccount** to user.

Table 3-2 ▪ FlexNet Operations Web Services Versions (cont.)

Web Service	Versions	Changes
User Account Hierarchy (continued)	**base**	The original User Account Hierarchy service introduced in FlexNet Operations 2018. It replaces the User Organization Hierarchy Service.
FlexNet Authentication	**v1**	Introduced in FlexNet Operations 2016 Release 4, v1 changes include ● Removal of deprecated API, **getUserToken()**.
	base	The original FlexNet Authentication service introduced in FlexNet Operations 2016.
Manage Device	**v7**	Introduced in FlexNet Operations 2025.05, v7 changes include: ● The **zeroCountAddonLineItems** operation enables producers to explicitly set an activation ID with a zero allocation mapping on a Cloud License Server (CLS) instance.
	v6	Introduced in FlexNet Operations 2023.08, v6 changes include: ● The **getDevice**, **getDevicesQuery** and **searchDevices** operations can now return the **SiteName** attribute value. ● The **getDevicesQuery** and **searchDevices** operations support the optional search parameter **SiteName**.
	v5	Introduced in FlexNet Operations 2022.11, v5 changes include: ● The **searchDevices** operation now supports searching and filtering by the parameters **addOnExpirationDate** and **hasLicense**.
	v4	Introduced in FlexNet Operations 2020 R3 SP3, v4 changes include: ● Added the **createdOn** attribute to **getDeviceResponseData**. This attribute enables users to view the device creation date.
	v3	Introduced in FlexNet Operations 2018 R1, v3 changes include: ● Changing **organization** to **account** and **org** to **acct**.
	v2	Introduced in FlexNet Operations 2017 R4, v2 changes include: ● Added the **soldToOrgId** attribute to **getDevicesQuery**. The **soldTo** attribute uses the display name when searching. The new parameter, **soldToOrgId**, uses the Account ID for searching. ● Added the **getAddonExpirationDate** service. This new service allows users to retrieve the expiration override date after an addon is mapped to a device.

Table 3-2 ▪ FlexNet Operations Web Services Versions (cont.)

Web Service	Versions	Changes
Manage Device (continued)	**v1**	Introduced in FlexNet Operations 2017, v1 changes include ● Introduction of a new User element for devices. ● Removal of unsupported operations.
	base	The original Manage Device service introduced in FlexNet Operations 2016.
Download Packaging	**v4**	Introduced in FlexNet Operations 2024.01, v4 changes include: ● Users can create and retrieve information about account and country restrictions for download packages: ● New endpoint **CreateRestrict** enables users to incorporate account and country restrictions or both into a download package. ● Existing endpoint **getDownloadPackageQuery** can now retrieve both types of restrictions. ● Added search parameter **AttributeDescriptor** to the **getDownloadPackageQuery** and **getDownloadPackageCount** methods to support querying download packages based on custom attribute values.
	v3	Introduced in FlexNet Operations 2022.10, v3 changes include: ● Added search parameter **LastModifiedDateTime** to the following operations: ● **getDownloadPackageQuery** ● **getDownloadPackageCount** ● **getFileQuery** ● **getFileCount** The new parameter enables producers to import data about periodic updates into their back office system and make this data available to their end customers. ● Added search parameter **AttributeDescriptor** to the **getDownloadPackageQuery** and **getDownloadPackageCount** methods to support querying download packages based on custom attribute values.

Table 3-2 ▪ FlexNet Operations Web Services Versions (cont.)

Web Service	Versions	Changes
Download Packaging (continued)	**v2**	Introduced in FlexNet Operations 2016 Release 4, v2 changes include ● The **getDownloadPackageQuery** is updated to accept product name and version to enable searching for a specified product via Web Services. The response is updated to include the system ID, version, and availability—ARCHIVE, CURRENT, or FUTURE—of the download package. ● To export download packages into Excel, product association information is included in the response of the **getDownloadPackageQuery** method. When an existing download package ID is passed and **ReturnContainedObjects** is set to TRUE, associated product names and the versions (if any) are returned per download package result. Similarly, to export download files into Excel, the download packages each file is associated with must be populated. This information is included in the response of the **getFileQuery** method. When an existing file ID is passed and **ReturnContainedObjects** is set to TRUE, associated download package IDs (if any) are returned per download file result.
	v1	Introduced in FlexNet Operations 2016 Release 3, v1 changes include ● Web Service versioning introduced. ● No functional changes.
	base	The original Electronic Software Delivery service introduced in FlexNet Operations 2016. This service was previously called ESD Web Service.
Usage	**v1**	Introduced in FlexNet Operations 2018 R1, v1 changes include: ● Changing **organization** to **account** and **org** to **acct**.
	base	The original Usage service introduced in FlexNet Operations 2016.
Electronic Software Delivery	**base**	The ESD service introduced in FlexNet Operations 2022.02.

SOAP Web Services Batch Limits

The following batch-size maximum limits apply for requests issued from SOAP web services:

- All read services have a maximum batch size of 2000.

- All services that result in a write have a batch size maximum of 25 records.

Specifying a batch size higher than the new cap results in an error.

Using FlexNet Operations SOAP Web Services

Any client application calling the FlexNet Operations Web Services needs the following:

- A Web Service client implementation containing stubs/proxies generated from the service interfaces
- An environment in which to develop and run clients to Web Services
- FlexNet user credentials (user ID and password) that have been assigned one or more roles with sufficient permissions to perform requested operations (See User for Executing SOAP Web Services.)
- A client program that you write to connect to the service that knows
    - The URL to connect to the service
    - FlexNet user name and password
    - The semantics of the service methods
    - The purpose of the each method
    - The return values expected from the methods

By appending `?wsdl` to the end of the Web Service URL, you obtain an automatically generated WSDL document that describes the Web Service:

```
https://<host:port>/flexnet/services/<serviceName>?wsdl
```

where `<serviceName>` is one of the base version Web Service names listed in Table 3-3. (To obtain a WSDL for a newer Web Service version, add the version designator before the service name. For example, `https://<host:port>/flexnet/services/v3/ManageDeviceService?wsdl`.)

Note that the location for the WSDL for the Download Packaging Service differs from that mentioned above:

```
https://<siteID>-fno.flexnetoperations.com/flexnet/services/<version>/DownloadPackagingService/wsdl/
DownloadPackagingService.wsdl.
```

Table 3-3 ▪ Names to Access FlexNet Operations Web Services

Web Service	Web Service Name	Parent Module
Product Packaging	`ProductPackagingService`	FlexNet Operations
Entitlement Order	`EntitlementOrderService`	FlexNet Operations
License	`LicenseService`	FlexNet Operations
User Account Hierarchy	`UserAcctHierarchyService`	FlexNet Advanced Organization Management
Flexnet Authentication	`FlexnetAuthentication`	FlexNet Operations

Table 3-3 ▪ Names to Access FlexNet Operations Web Services (cont.)

Web Service	Web Service Name	Parent Module
Manage Device	`ManageDeviceService`	FlexNet Advanced Lifecycle Management
Download Packaging	`DownloadPackagingService`	Software Delivery
Usage	`UsageService`	FlexNet Usage

Authorizing Calls to SOAP Web Services

The following authorization methods are currently available:

- Token-based authorization—for more information, see the chapter Token-Based Authentication.
- Basic HTTP user authentication—for more information, see the following section, User for Executing SOAP Web Services.

User for Executing SOAP Web Services

A client connecting to a FlexNet Operations Web Service must provide the user name and password of a user authorized to

- Execute FlexNet Operations Web Services
- Perform the requested task, the same as if the user were logged into the FlexNet Operations user interface

Creating a SOAP Web Service User

Though Web Services can be tested using the built-in **System Administration** user (ADMNadmin), Web Services should not be run as this user in production. Create one or more FlexNet users who have been assigned roles with the necessary permissions.

Creating a Role for Web Service User Access

Task ***To create a role that provides access to Web Services***

1. Start the FlexNet Operations Producer Portal and log in as a user with a Producer role.
2. Click **Accounts & Users** > **All Roles** > **Add Role**.
3. Create the role as usual. Under **Role Type**, select **Publisher Role**.
4. Check the **Execute Web Services** permission.
5. Check any other permissions required to perform the tasks required by the Web Services operations performed by users with this role.
6. Click **Save**.

After the appropriate roles have been created, you must create a FlexNet Web Service user.

Creating a Web Services User for FlexNet Operations

Follow the instructions, below, to create a Web Service user.

Task ***To create a FlexNet Web Service user***

1. In the FlexNet Operations Producer Portal, click **Accounts & Users** > **Create User**.
    - If the Web Service user is in the FlexNet domain, enter a user name (login name), first and last names, and email address of the user. Leave the account the Home account. Select the time zone of the FlexNet Operations server.
    - If the Web Service user is in a domain other than the FlexNet domain, select that domain select the user from the domain.
    - If this is a user in the FlexNet domain and it will ever send more than one Web Service request at a time, in the **Shared Login** field, select **Yes**. If this user is created as a shared user, enter the email address of a FlexNet Operations administrator.
2. Assign the Web Services role created in the previous task to the user.
3. Click **Save**.

If the user is in the FlexNet domain, an email with a system-generated password is sent to the user's email address. The recipient of the email must log in to FlexNet Operations with the system-generated password and change that password. A user in a domain other than the FlexNet domain uses the password managed in the directory server.

***Note** ▪ Note the changed password and keep it secure—it is used to authenticate the user in the Web Services requests.*

Passing the SOAP Web Service User

By default, FlexNet Operations Web Services requires a client to provide credentials as BASIC HTTP user authentication. BASIC authentication encrypts the user ID and password with Base64 encoding and passes it as an HTTP request header parameter. For example:

```
Authorization: Basic QWxhghwelhijhrcGVuIHNlc2FtZQ==
```

However, not all Web Services client tools offer the ability to pass custom HTTP request header parameters in the request. Therefore, FlexNet Operations supports an alternative mechanism to pass user credentials as SOAP header parameters along with the SOAP message. The two case-sensitive parameters are `UserId` and `Password`. The password must be Base64 encoded.

```
<soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <flex:UserId soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
        soapenv:mustUnderstand="0" xmlns:flex="urn:com.macrovision:
        flexnet/platform">admin</flex:UserId>
   <flex:UserPassword soapenv:actor="http://schemas.xmlsoap.org/soap/actor/
        next"soapenv:mustUnderstand="0" xmlns:flex="urn:com.macrovision:
```

```
        flexnet/platform">YWRtaW4=</flex:UserPassword>
</soapenv:Header>
```

If the user is authenticated in a domain other than the FlexNet domain, the user ID must be passed as *`userName##domain`*. *`domain`* is the name specified when the domain was added to FlexNet Operations. This syntax is also valid for users in the FlexNet domain. For example, `admin##FLEXnet`.

Choosing a SOAP Web Service Client Implementation

Choose one of these types of client implementations to write client applications to connect to the Web Services:

- Create a Basic Web Service Client in C# (see Creating a Basic Web Services Client in C Sharp Using .NET and Creating a Basic Web Services Client in C Sharp Using ASP.NET Core)
- Generate your own Axis-Java client implementation (see Generating Your Own Axis–Java Client Implementation)
- Generate a client implementation with a different tool

Run your client applications using the same environment that you use to create the client implementation.

Creating a Basic Web Services Client in C Sharp Using .NET

To create a simple Entitlement using the EntitlementOrderService wsdl, you will need to create a C# Console Application project with a Web Reference that contains an override function which specifies that requests are to use "Basic" HTTPS authentication, adding a new “Authorization” header with credentials of the form "Basic " plus the base-64-encoded "Name:Password" string.

Task ***Creating a Simple Entitlement for .NET***

1. In Visual Studio 2008 or later, create a new C# Console Application project.
2. In the **Solution Explorer**, right-click the project icon and select **Add Service Reference**.
3. Click **Advanced** at the bottom of the **Add Service** screen and then **Add Web Reference** at the bottom of the **Service Reference Settings** screen.
4. Enter **`https://<host:port>/flexnet/services/EntitlementOrderService?wsdl`** and click **Go**. Alternatively, you can enter **`https://<host:port>/flexnet/services`** to see a list of all available FlexNet Operations WSDLs.
5. Change the Web reference name to **`EntOrdSvc`** and click **Add Reference**. The **Solution Explorer** view should have a new **EntOrdSvc** icon in the **Web References** folder.
6. Double-click the **EntOrdSvc** icon, which opens the **Object Browser**.
7. In the **Object Browser**, expand **WebServiceTest.EntOrdSvc**, and double-click one of the items in the tree. This should open the proxy class source file `Reference.cs`.
8. Near the top of `Reference.cs`, find the line that begins:

```
public EntitlementOrderService()
```

and insert the following function directly above:

```
protected override WebRequest GetWebRequest(Uri uri)
{
    HttpWebRequest request;
    request = (HttpWebRequest)base.GetWebRequest(uri);

    if (PreAuthenticate)
    {
        NetworkCredential networkCredentials =
            Credentials.GetCredential(uri, "Basic");
        if (networkCredentials != null)
        {
            byte[] credentialBuffer =
            new
            System.Text.UTF8Encoding( ).GetBytes(
                networkCredentials.UserName + ":" +
                networkCredentials.Password);

            request.Headers["Authorization"] =
                "Basic " +
                Convert.ToBase64String(credentialBuffer);
        }
        else
        {
            throw new ApplicationException(
                "No network credentials");
        }
    }
    return request;
}
```

9. At the top of the file in the namespace function for the project, add the following:

```
using System.Net;
```

10. In the `Program.cs` file, add a Web Service request to create an entitlement. For example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using WebServiceTest.EntOrdSvc;

namespace WebServiceTest
{
    class Program
    {
        static void Main(string[] args)
        {
            EntitlementOrderService eos = new EntitlementOrderService();

            // begin copied-and-pasted authorization code
            NetworkCredential netCredential =
                new NetworkCredential("username@company.com", "MyPassword");
```

```
Uri uri = new Uri(eos.Url);

ICredentials credentials = netCredential.GetCredential(uri, "Basic");

eos.Credentials = credentials;
eos.PreAuthenticate = true;

Console.WriteLine("[Successfully authenticated...]");
// end copied-and-pasted authorization code

// one and only simple entitlement
createSimpleEntitlementDataType[] seArray = new
    createSimpleEntitlementDataType[1];
createSimpleEntitlementDataType se1 = new createSimpleEntitlementDataType();
seArray[0] = se1;

idType idtype = new idType();
// make sure to change the Order ID for each test...
idtype.id = "ExampleOrderID";
se1.entitlementId = idtype;
// make sure this is a real customer account
se1.soldTo = "Atlas";

// one and only line item
createEntitlementLineItemDataType[] lineItems = new
    createEntitlementLineItemDataType[1];
se1.lineItems = lineItems;
lineItems[0] = new createEntitlementLineItemDataType();
idtype = new idType();
idtype.id = "ActID-Atlas-123456";
lineItems[0].activationId = idtype;
lineItems[0].numberOfCopies = "5";
lineItems[0].startDate = DateTime.Today;

licenseModelIdentifierType lm = new licenseModelIdentifierType();
licenseModelPKType modelPk = new licenseModelPKType();

// Add a License Model for the desired Product
modelPk.name = "Embedded Counted";
lm.primaryKeys = modelPk;
lineItems[0].licenseModel = lm;
lineItems[0].versionDate = DateTime.Today.Date;

productIdentifierType ordId = new productIdentifierType();
productPKType productPk = new productPKType();
// Be sure to pick a real product and version
productPk.name = "LH Full Access";
productPk.version = "1.0";
ordId.primaryKeys = productPk;
lineItems[0].product = ordId;

// this function is in the auto-generated Reference.cs code
createSimpleEntitlementResponseType csert = eos.createSimpleEntitlement(seArray);

if (csert.statusInfo.status != StatusType.SUCCESS)
{
```

```
                    Console.WriteLine("Drat! Failed: " +
                        csert.statusInfo.reason);
                }
                else
                {
                    Console.WriteLine("Success!");
                }
            }
        }
    }
```

11. Build and run the application.

Creating a Basic Web Services Client in C Sharp Using ASP.NET Core

To create a simple Entitlement using the EntitlementOrderService wsdl, you will need to create a C# Console Application project with a Web Reference that contains an override function which specifies that requests are to use "Basic" HTTPS authentication, adding a new "Authorization" header with credentials of the form "Basic " plus the base-64-encoded "Name:Password" string.

Task ***Creating a Simple Entitlement for ASP.NET Core***

1. In Visual Studio 2022 or later, create a new ASP.NET CORE WEB API project.
2. In the **Solution Explorer**, right-click the project icon and select **Add** > **Service Reference**.

   Alternatively, you can also open the **Connected Services** and click **Add a service reference**.
3. Select **WCF Web Service** and click **Next**.
4. In the **URI** field, enter `https://<host:port>/flexnet/services/EntitlementOrderService?wsdl` and click **Go**.

   Alternatively, enter `https://<host:port>/flexnet/services` to see a list of all available FlexNet Operations WSDLs.
5. Choose the service listed in the **Services** section to view the available APIs/operations.
6. Change the **Namespace** to `EntitlementOrderServiceReference` and click **Next**.
7. In the **Specify the client options** screen, select the check boxes **Generate Synchronous Operations** and **Enable Data Binding**. This allows you to generate both asynchronous and synchronous client operations.
8. Click **Finish**. The **Solution Explorer** window now shows the new **EntitlementOrderService** icon in the **Connected Services** folder.
9. Double-click the **EntitlementOrderService** icon, which opens the **Object Browser**.
10. Open the `Reference.cs` proxy class source file.
11. Search for the **EntitlementOrderServiceInterfaceV5Client** class and add the following code above it:

```
/// <summary>
/// Add credentials and endpoint headers here
/// </summary>
```

```
public partial class EntitlementOrderServiceInterfaceV5Client :
    System.ServiceModel.ClientBase<EntitlementOrderServiceReference.
    EntitlementOrderServiceInterfaceV5>,
    sEntitlementOrderServiceReference.EntitlementOrderServiceInterfaceV5
{
    static partial void ConfigureEndpoint(System.ServiceModel.Description.ServiceEndpoint
    serviceEndpoint, System.ServiceModel.Description.ClientCredentials clientCredentials)
    {
        byte[] credentialBuffer = new System.Text.UTF8Encoding().GetBytes("***Username***" + ":" +
        "***Password***");
        var headers = new Dictionary<string, string>
            {
                {"Authorization", "Basic " + Convert.ToBase64String(credentialBuffer)}
            };
        var behavior = new AddHttpHeaderMessageEndpointBehavior(headers);
        serviceEndpoint.EndpointBehaviors.Add(behavior);

        clientCredentials.UserName.UserName = "***Username***";
        clientCredentials.UserName.Password = "***Password***";
    }
}
    /// <summary>
    /// define custom endpoint behavior
    /// </summary>
    public class AddHttpHeaderMessageEndpointBehavior : IEndpointBehavior
    {
        private readonly IClientMessageInspector _httpHeaderMessageInspector;

        public AddHttpHeaderMessageEndpointBehavior(Dictionary<string, string> headers)
        {
            _httpHeaderMessageInspector = new HttpHeaderMessageInspector(headers);
        }

        public void AddBindingParameters(ServiceEndpoint endpoint, BindingParameterCollection
            bindingParameters)
        {

        }

        public void ApplyClientBehavior(ServiceEndpoint endpoint, ClientRuntime clientRuntime)
        {
            clientRuntime.ClientMessageInspectors.Add(_httpHeaderMessageInspector);
        }

        public void ApplyDispatchBehavior(ServiceEndpoint endpoint, EndpointDispatcher
            endpointDispatcher)
        {

        }

        public void Validate(ServiceEndpoint endpoint)
        {

        }
    }
```

```
/// <summary>
/// define customer message inspector
/// </summary>
public class HttpHeaderMessageInspector : IClientMessageInspector
{
    private readonly Dictionary<string, string> _headers;

    public HttpHeaderMessageInspector(Dictionary<string, string> headers)
    {
        _headers = headers;
    }

    public void AfterReceiveReply(ref Message reply, object correlationState)
    {

    }

    public object BeforeSendRequest(ref Message request, IClientChannel channel)
    {
        if (request.Properties.Count == 0 || request.Properties
            [HttpRequestMessageProperty.Name] == null)
        {
            request.Properties.Add(HttpRequestMessageProperty.Name, new
                HttpRequestMessageProperty());
        }
        var headersCollection =
            ((HttpRequestMessageProperty)request.Properties[HttpRequestMessageProperty.Name])
                .Headers;

        foreach (var header in _headers) headersCollection[header.Key] = header.Value;

        return null;
    }
}
```

12. Search for all occurrences of `***Username***` and `***Password***`, using the correct credentials.
13. Search for the `GetBindingForEndpoint` method definition and replace it with the following code:

```
private static System.ServiceModel.Channels.Binding GetBindingForEndpoint(EndpointConfiguration
endpointConfiguration)
{
    if ((endpointConfiguration == EndpointConfiguration.EntitlementOrderServiceV5))
    {
        System.ServiceModel.BasicHttpsBinding result = new System.ServiceModel.BasicHttpsBinding();
        result.Security.Transport.ClientCredentialType = HttpClientCredentialType.Basic;
        result.MaxBufferSize = int.MaxValue;
        result.ReaderQuotas = System.Xml.XmlDictionaryReaderQuotas.Max;
        result.MaxReceivedMessageSize = int.MaxValue;
        result.AllowCookies = true;
        return result;
    }
    throw new System.InvalidOperationException(string.Format("Could not find endpoint with name
        \'{0}\'.", endpointConfiguration));
}
```

14. Search for the **`GetEndpointAddress`** method definition and change the URL used to return the `EndpointAddress` from `http` to **`https`**.

15. Create a new Controller **`"EntitlementOrderController"`**. Replace the code generated by the Visual Studio editor with the following code:

```
using EntitlementOrderServiceReference;
using Microsoft.AspNetCore.Mvc;

namespace ConsumeWSDLConnectedServices.Controllers
{
    [ApiController]
    [Route("EntitlementOrderController")]
    public class EntitlementOrderController : Controller
    {
        [Route("CreateSimpleEntitlement")]
        [HttpPost]
        public ActionResult CreateSimpleEntitlement()
        {
            EntitlementOrderServiceInterfaceV5Client entitlementOrderServiceInterface =
                new EntitlementOrderServiceInterfaceV5Client();

            // one and only simple entitlement
            createSimpleEntitlementRequestType createSimpleEntitlementRequestType =
                new createSimpleEntitlementRequestType();
            createSimpleEntitlementDataType[] seArray = new createSimpleEntitlementDataType[1];
            createSimpleEntitlementDataType se1 = new createSimpleEntitlementDataType();

            seArray[0] = se1;
            idType idtype = new idType();

            // make sure to change the Order ID for each test...

            idtype.id = "ExampleOrderID";
            se1.entitlementId = idtype;

            // make sure this is a real customer account

            se1.soldTo = "Atlas";

            // one and only line item

            createEntitlementLineItemDataType[] lineItems =
                new createEntitlementLineItemDataType[1];
            se1.lineItems = lineItems;
            lineItems[0] = new createEntitlementLineItemDataType();
            idtype = new idType();
            idtype.id = "ActID-Atlas-123456";
            lineItems[0].activationId = idtype;
            lineItems[0].numberOfCopies = "5";
            lineItems[0].startDateSpecified = true;
            lineItems[0].startDate = DateTime.Today;
            lineItems[0].expirationDateSpecified = true;
            lineItems[0].expirationDate = DateTime.Today.AddYears(1);

            licenseModelIdentifierType lm = new licenseModelIdentifierType();
```

```
                licenseModelPKType modelPk = new licenseModelPKType();

                // Add a License Model for the desired Product

                modelPk.name = "Embedded Counted";
                lm.primaryKeys = modelPk;
                lineItems[0].licenseModel = lm;
                lineItems[0].versionDate = DateTime.Today.Date;

                productIdentifierType ordId = new productIdentifierType();
                productPKType productPk = new productPKType();

                // Be sure to pick a real product and version

                productPk.name = "LH Full Access";
                productPk.version = "1.0";
                ordId.primaryKeys = productPk;
                lineItems[0].product = ordId;

                // this function is in the auto-generated Reference.cs code

                createSimpleEntitlementRequestType.simpleEntitlement = seArray;

                createSimpleEntitlementResponseType csert =
                    entitlementOrderServiceInterface.createSimpleEntitlement
                    (createSimpleEntitlementRequestType);

                if (csert.statusInfo.status != StatusType.SUCCESS)
                {
                    Console.WriteLine("Drat! Failed: " + csert.statusInfo.reason);
                }
                else
                {
                    Console.WriteLine("Success!");
                }

                return new JsonResult(csert);
            }
        }
    }
```

16. Build and run the application.

Generating Your Own Axis–Java Client Implementation

If you want a Axis–Java implementation that reflects the Java package structure of your own, you can generate your own Axis–Java implementation from the Web Service WSDLs using the WSDL2Java utility in Axis. This has been tested using Axis 1.4 and JDK 1.8.x.

Product Packaging Service

Information about the products you develop and sell may be stored in an Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) system. The Product Web Service facilitates the integration of your ERP or CRM system with FlexNet Operations. It can be used to import product definitions and to import part numbers that can be associated with products you create through the FlexNet Operations user interface or Product Packaging Service. This section describes the ProductPackagingService interface.

- ProductPackagingService Interface presents the XML files that define the ProductPackagingService interface. The interface is used to generate language-specific client implementations that can communicate with the Product Packaging Service to import a product hierarchy.
- Product Packaging Service Operations contains a list of the basic types of Web Services operations.
- Using the Product Packaging Service Operations gives an overview of how to use the Product Packaging Service operations.

ProductPackagingService Interface

The endpoint to connect to the Product Packaging service depends on the Web Service version you are using:

- **Default endpoint**—`https://<host:port>/flexnet/services/ProductPackagingService`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/ProductPackagingService`

`ProductPackagingService.wsdl` defines the ProductPackagingService interface and references `productPackagingTypes.xsd`, which in turn references the XML Schema documents that define Product Packaging Service custom data types, service requests, and service responses.

Product Packaging Service Operations

The following table shows the basic types of Web Services operations that manage the entities in the product hierarchy: features, feature bundles, products, suites, and maintenance. Many of these types of operations are also supported in the other FlexNet Operations Web Services.

Table 3-4 ▪ Patterns in Product Packaging Service Operations

Operation Type	Description
create	Create the entity in the draft state.
update	Update an existing entity with the new values provided, if its state allows modification. (Depending on the state and the modified field this may require the **Allow Editing of Deployed Entities** setting to be checked on the Producer Portal—**System > Configure > FlexNet Operations**.)
delete	Delete an existing entity, if its state allows deletion.
getCount	Return the number of entities that match the specified criteria.
getQuery	Return the entities that match the specified query parameters.

Table 3-4 ▪ Patterns in Product Packaging Service Operations (cont.)

Operation Type	Description
get	Provide one query parameter (usually an identifier) and return data.
setState	Change the state of an entity, if its current state allows the change.

Using the Product Packaging Service Operations

Before using FlexNet Operations Web Services, plan how to exchange customer, product, order, and fulfillment information between FlexNet Operations and your other back-office systems. The following table provides a quick overview of how to approach managing your product hierarchy with the Product Packaging Service.

Table 3-5 ▪ Product Packaging Service Operations

Step	Description
Step 1	Plan how you want to license, package, and support the lifecycle of licenses for your products.
Step 2	Create a license generator configuration for your production VCG or custom license technologies, custom license generator configurations, and custom attributes, if necessary. These entities cannot be created or managed through Web Services.
Step 3	Create license models (and transaction keys, if you are using trusted license models) in the FlexNet Operations UI. These entities cannot be created or managed through Web Services. In additional to deciding how to express your licensing strategy in one or more license models, consider when and by whom you want values for license attributes to be set—that is, by a product manager at the time a license model is created, by an order entry person in your company, by a support person in your company who generates licenses with input provided by customers, or by the customers themselves with the End-User Portal. Also consider policies on license lifecycle operations such as extra activations, returns, and rehosts, especially if your customers will use the End-User Portal.
Step 4	If you use part numbers to identify your products, import part numbers using the Product Packaging Service.
Step 5	Create features and optionally group features into convenient feature bundles.
Step 6	Create maintenance products to represent the value of your support contracts.
Step 7	Create products by adding features and/or feature bundles, linking license models and possibly a transaction key, mapping part numbers, and relating products to maintenance and other products. You may want to use the two getIdentifiers operations to populate look-up tables of the available transaction keys and license model names.
Step 8	Optionally, you can create suites containing groups of products.
Step 9	Create accounts.

Table 3-5 ▪ Product Packaging Service Operations

Step	Description
Step 10	If you want to test license models and products while you can modify them, you are ready to create test entitlements. Before you can create deployable entitlements that can be fulfilled by customers, you must deploy your license models (only through the UI) and product hierarchy.

Entitlement Order Service

The Entitlement Order service facilitates the integration of your ERP/CRM system with FlexNet Operations by creating regular and bulk entitlements. This section describes the EntitlementOrderService interface.

- EntitlementOrderService Interface presents the XML files that define the EntitlementOrderService interface. The interface is used to generate language-specific client implementations that can communicate with the Entitlement Order service to import entitlements.
- Specifying a Product and License Model when Creating a Line Item or Bulk Entitlement explains how to define at least one product and one primary license model to deploy an entitlement line item or bulk entitlement.
- Using the Entitlement Order Service Operations gives an overview of how to use the Entitlement Order service operations.

EntitlementOrderService Interface

The endpoint to connect to the Entitlement Order service depends on the Web Service version you are using:

- **Default endpoint with SSL**—`https://<host:port>/flexnet/services/EntitlementOrderService`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/EntitlementOrderService`

`EntitlementOrderService.wsdl` defines the EntitlementOrderService interface and references `entitlementOrderTypes.xsd`, which in turn references the XML Schema documents that define Entitlement Order service custom data types, service requests, and service responses.

Specifying a Product and License Model when Creating a Line Item or Bulk Entitlement

To deploy an entitlement line item or bulk entitlement, at least one product and one primary license model must be defined. The contents of Web Service requests that create entitlements may or may not specify product and license model data explicitly, but between the request contents and the product data in FlexNet Operations, there must be enough information provided to uniquely define at least one product and one primary license model on the line item or bulk entitlement. You can define what is necessary in several ways, depending on the following factors:

- The number of license models linked to a product
- The mapping of part numbers to products

- Are part numbers mapped to products?
- How many part numbers are mapped to each product—one to one or many to one?

A part number can be mapped either to a particular product-license model pair or just to a product. Each part number can be mapped to only one product, but one product can have many part numbers mapped to it.

There are two primary ways to specify a product and license model:

1. Use the `product` element (preserved for backward compatibility) or no element to specify a single product.
2. Use the `entitledProducts` element to specify one or more products.

The `entitledProducts` element includes one or more `entitledProduct` elements, and each `entitledProduct` element contains a `product` element and a `quantity`. In the request, it is possible to use either a single `product` element without `entitledProducts` or to use an `entitledProducts` element that contains one or more products, each encapsulated in an `entitledProduct`. Use either the `product` element or the `entitledProducts` element in the request, not both.

***Tip** ▪ For update operations with multiple products, use `updatedEntitledProducts` instead of `entitledProducts`.*

Using the product Element or No Element to Specify a Product

For Entitlement Order Service requests that create line items or bulk entitlements, if there are no elements that specify a product or if a `product` element specifies a product, FlexNet Operations first looks for a part number in the request. If a part number is specified in the Web Service request, the part number is used to identify the product in the request. If a part number is omitted from the request, FlexNet Operations tries to get the product from the request. The following sections discuss how FlexNet Operations behaves when a part number is specified in the request versus when a part number is not specified.

Part Number Specified

If a part number is provided in the request, FlexNet Operations attempts to determine the product based on the part number. The following points describe the FlexNet Operations behavior when a request includes a part number.

Table 3-6 ▪ Request Contains a Product and a Part Number

Conditions	Results
The part number is not mapped to an existing product.	An error is generated.
The part number is mapped to a particular license model linked to a product.	The part number is used to identify both the product and the license model in the request. (If a product or a license model is also specified in the request, it is ignored.)
The part number is mapped to a product with exactly one license model linked to it.	The license model does not have to be specified in the request.

Table 3-6 ▪ Request Contains a Product and a Part Number (cont.)

Conditions	Results
The part number is mapped to a product with more than one license model.	FlexNet Operations tries to get the license model from the request. If no license model is specified in the request, and the line item or bulk entitlement is set to be deployed in the request, an error is generated.

Part Number not Specified

If a part number is not included in the request, FlexNet Operations tries to get the product from the request. The following points describe the FlexNet Operations behavior when a request excludes a part number.

Table 3-7 ▪ Request Contains a Product and No Part Number

Conditions	Results
No product exists in the request.	FlexNet Operations generates an error.
The product in the request is linked to exactly one license model.	The linked license model is used. (Any license model specified in the request is ignored.)
The product in the request is linked to multiple license models.	FlexNet Operations tries to get the license model from the request. If there is no license model in the request, and the line item or bulk entitlement is set to be deployed, an error is generated.
The product in the request is mapped to one or more part numbers (but the part numbers are not mapped to the license model specified in the request).	No part number is loaded into the entitlement.
Both the product and license model in the request are mapped to one or more part numbers.	One of the part numbers (indeterminate) is loaded into the entitlement. Therefore, if part numbers are mapped to products, it is a best practice to specify a part number in Web Service requests.

Using the entitledProducts Element

For Entitlement Order Service requests that create line items or bulk entitlements, if an `entitledProducts` element is used, it might contain multiple products. Therefore, if a part number is specified along with the `entitledProducts` element, the part number is not used to specify a product. (A part number can be mapped to only one product.) If a part number is specified in the request, FlexNet Operations checks that the part number is mapped to one of the products specified in the `entitledProducts` element.

Part Number Specified

If a part number is provided in the request, FlexNet Operations tries to get the license model from the request.

Table 3-8 ▪ Request Contains entitledProducts and a Part Number

Conditions	Results
The part number is not mapped to a product specified in the `entitledProducts` element.	An error is generated.
Part number maps to a license model that is linked to one or more products in the `entitledProducts` element.	FlexNet Operations uses the part number to identify the license model for the request. (License models specified in the request are ignored.)
● Part number maps to one of the products in the `entitledProducts` element. ● Mapped product is linked with exactly one license model.	FlexNet Operations uses the license model linked with the products. (License models specified in the request are ignored.)
● Part number maps to one of the products in the `entitledProducts` element. ● Mapped product is linked to multiple license models.	FlexNet Operations tries to get the license model from the request. If no license model is specified in the request, an error is generated.

Part Number not Specified

If a part number is omitted from the request, FlexNet Operations tries to get the products and license models from the request. Even if exactly one license model is linked to all the products in the `entitledProducts` element, FlexNet Operations still tries to get the license model from the request.

Table 3-9 ▪ Request Contains entitledProducts and No Part Number

Conditions	Results
No license model exists in the request.	An error is generated.
Multiple part numbers are mapped to the product-license model combinations specified in the request.	One of the part numbers (indeterminate) is loaded into the entitlement. Therefore, if part numbers are mapped to products, it is best to specify a part number in the Web Service request.

The following table shows the operations that create line items or bulk entitlements and use data types that allow part numbers to specify products or products and license models:

Table 3-10 ▪ Data Types that Find Products from Part Numbers

Web Service Operation	Data Type
createSimpleEntitlement	createEntitlementLineItemDataType createMaintenanceLineItemDataType
createEntitlementLineItem	createEntitlementLineItemDataType createMaintenanceLineItemDataType
replaceEntitlementLineItem	createEntitlementLineItemDataType createMaintenanceLineItemDataType
createBulkEntitlement	createBulkEntitlementDataType
upgradeLicense	createEntitlementLineItemDataType
upsellLicense	createEntitlementLineItemDataType

Using the Entitlement Order Service Operations

The following provides a quick overview of how to approach managing entitlements with the Entitlement Order service. These steps assume that you already have license models, the product hierarchy, and accounts in your FlexNet Operations database.

Table 3-11 ▪ Steps for Managing Entitlements

Step	Description
Step 1	Plan how you will transform orders into entitlements.
Step 2	Do you map part numbers to your products with a single license model, with multiple license models, or to individual product-license model combinations? Consider identifying products or products and license models in entitlement line items with part numbers.
Step 3	When you have identified a license model to be used in an entitlement line item, identify the license model attributes whose values must be set at entitlement time using the getEntitlementAttributesFromModel operation.
Step 4	If you want to test license models and the product hierarchy while you can still modify them, create test entitlements. When test entitlements are deleted, fulfillments created from them are also deleted.
Step 5	After you have deployed license models and your product hierarchy, create deployable simple or bulk entitlements for your customers. Add entitlement line items (identified by activation IDs) and provide values for entitlement-time license attributes.

Table 3-11 ▪ Steps for Managing Entitlements

Step	Description
Step 6	Load web register keys (activation IDs) into bulk entitlements.
Step 7	Optionally, send your customers entitlement certificates listing the products to which the customers are entitled and their activation IDs.
Step 8	Consider using getEntitlementsQuery, getActivatableItemCount, getActivatableItemsQuery, getBulkEntitlementPropertiesQuery and getEntitlementLineItemPropertiesQuery operations to support your license fulfillment process.
Step 9	Optionally, renew, upgrade, or upsell entitlements. Upgrading and upselling require that upgrade or upsell product relationships between products have been configured.

License Service

The License Web Service allows you to generate and email certificate licenses, perform license lifecycle operations for FlexNet Licensing certificate and custom technology licenses, and generate short code activation responses for FlexNet Licensing trusted activations. This section describes the LicenseService interface.

- LicenseService Interface presents the XML files that define the LicenseService interface. The interface is used to generate language-specific client implementations that can communicate with the License service to import entitlements.
- Using the License Service Operations gives an overview of how to use the License service operations.

LicenseService Interface

The endpoint to connect to the License service depends on the Web Service version you are using:

- **Default endpoint**—`https://<host:port>/flexnet/services/LicenseService`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/LicenseService`

`LicenseFulfillmentService.wsdl` defines the LicenseService interface and references `licenseFulfillmentTypes.xsd`, which in turn, references the XML Schema documents that define License service custom data types, service requests, and service responses.

Using the License Service Operations

The following provides a quick overview of how to approach managing fulfillments with the License service. These steps assume that you have planned how your customers will fulfill their licenses and whether and how you support license lifecycle operations.

Table 3-12 ▪ License Service Operations

Step	Description
Step 1	If you want to test license models and the product hierarchy while you can still modify them, create test entitlements and generate licenses for them. When test entitlements are deleted, fulfillments created from them are also deleted.
Step 2	After you have deployed license models and your product hierarchy, create deployable simple or bulk entitlements for your customers. Load web register keys (activation IDs) into bulk entitlements.
Step 3	Consider using getEntitlementsQuery, getActivatableItemCount, getActivatableItemsQuery, getBulkEntitlementPropertiesQuery and getEntitlementLineItemPropertiesQuery operations to support your license fulfillment process.
Step 4	When you have chosen activatable items, identify the license model attributes whose values must be set at fulfillment time using the getLicenseModelIdentifiers operation in the Product Packaging Service (required counts and hostids) and the **getFulfillmentAttributesFromModel** or **getFulfillmentAttributesForBatch** operation.
Step 5	Verify that licenses can be generated with the supplied attribute values, then generate licenses. Decide whether to override any policy you have set on the license model concerning extra activations.
Step 6	Optionally, consolidate fulfillments.
Step 7	Deliver licenses or consolidated licenses as email attachments.
Step 8	If you support the license lifecycle operations, process license renewals, upgrades, upsells, returns, repairs, and rehosts. Decide whether to override any policy you have set on the license model concerning lifecycle operations.

User Account Hierarchy Service

The User Account Hierarchy service creates, updates, and queries account hierarchies and creates, updates, or deletes users and user roles in an account. This section describes the UserAcctHierarchyService interface.

- UserAcctHierarchyService Interface presents the XML files that define the UserAcctHierarchyService interface. The interface is used to generate language-specific client implementations that can communicate with the User Account Hierarchy service to manage accounts and users.
- Creating an Account for .NET provides example code for creating an account using the UserAcctHierarchyService wsdl.

***Version* ▪ As of the FlexNet Operations 2017 R1 release, UserAcctHierarchyService replaces the UserOrgHierarchyService.**

UserAcctHierarchyService Interface

The endpoint to connect to the User Account Hierarchy service depends on the Web Service version you are using:

- **Default endpoint**—`https://<host:port>/flexnet/services/UserAcctHierarchyService`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/UserAcctHierarchyService`

`UserAcctHierarchyService.wsdl` defines the User Account Hierarchy interface and references `UserAcctHierarchyService.xsd` that defines User Account Hierarchy service custom data types, service requests, and service responses.

Creating an Account for .NET

To create an account using the UserAcctHierarchyService wsdl, you will need to create a C# Console Application project with a Web Reference that contains an override function which specifies that requests are to use "Basic" HTTPS authentication.

Task ***Creating an account for .NET***

1. In Visual Studio 2008 or later, create a new C# Console Application project.
2. In the **Solution Explorer**, right-click the project icon and select **Add Service Reference**.
3. Click **Advanced** at the bottom of the **Add Service** screen and then **Add Web Reference** at the bottom of the **Service Reference Settings** screen.
4. Enter **`https://<host:port>/flexnet/services/v3/UserAcctHierarchyService?wsdl`** and click **Go**. Alternatively, you can enter **`https://<host:port>/flexnet/services`** to see a list of all available FlexNet Operations WSDLs.
5. Change the Web reference name to **`UserAcctOrdSvc`** and click **Add Reference**. The **Solution Explorer** view should have a new **UserAcctOrdSvc** icon in the **Web References** folder.
6. Double-click the **UserAcctOrdSvc** icon, which opens the **Object Browser**.
7. In the **Object Browser**, expand **WebServiceTest.UserAcctOrdSvc**, and double-click one of the items in the tree. This should open the proxy class source file `Reference.cs`.
8. Near the top of `Reference.cs`, find the line that begins:

   ```
   public v3UserAcctHierarchyService()
   ```

   and insert the following function directly above:

   ```
   protected override WebRequest GetWebRequest(Uri uri)
   {
       HttpWebRequest request;
       request = (HttpWebRequest)base.GetWebRequest(uri);
   ```

```
        if (PreAuthenticate)
        {
            NetworkCredential networkCredentials =
                Credentials.GetCredential(uri, "Basic");
            if (networkCredentials != null)
            {
                byte[] credentialBuffer =
                new
                System.Text.UTF8Encoding( ).GetBytes(
                    networkCredentials.UserName + ":" +
                    networkCredentials.Password);

                request.Headers["Authorization"] =
                    "Basic " +
                    Convert.ToBase64String(credentialBuffer);
            }
            else
            {
                throw new ApplicationException(
                    "No network credentials");
            }
        }
        return request;
    }
```

9. At the top of the file in the namespace function for the project, add the following:

```
using System.Net;
```

10. In the `Program.cs` file, add a Web Service request to create an account. For example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using WebServiceTest.UserAcctOrdSvc;

namespace WebServiceTest
{
    class Program
    {
        static void Main(string[] args)
        {
            v3UserAcctHierarchyService usrAccSer = new
                v3UserAcctHierarchyService();

            // begin copied-and-pasted authorization code
            NetworkCredential netCredential =
                new NetworkCredential("username@company.com", "MyPassword");
            Uri uri = new Uri(usrAccSer.Url);

            ICredentials credentials = netCredential.GetCredential(uri, "Basic");

            usrAccSer.Credentials = credentials;
            usrAccSer.PreAuthenticate = true;

            Console.WriteLine("[Successfully authenticated...]");
```

```
                // end copied-and-pasted authorization code

                // one and only account
                createAcctRequestType acc1 = new createAcctRequestType();

                accountDataType[] account = new accountDataType[1];
                acc1.account = account;
                account[0] = new accountDataType();

                // make sure to change the Account ID for each test...

                account[0].id = "ExampleAccountID";
                account[0].name = "ExampleAccountName";
                account[0].description = "ExampleDescriptionForAccount";

                // add address for the respective account

                addressDataType address = new addressDataType();

                address.address1 = "No.45, 2nd Street";
                address.address2 = "Sunset Avenue";
                address.city = "Bangalore";
                address.state = "Karnataka";
                address.zipcode = "560054";
                address.country = "IN";
                address.region = "Asia";

                AcctType accountType = new AcctType();
                accountType = AcctType.CUSTOMER;

                // this function is in the auto-generated Reference.cs code
                createAcctResponseType creatAccResType = usrAccSer.createAccount(acc1);

                if (creatAccResType.statusInfo.status != StatusType.SUCCESS)
                {
                    Console.WriteLine("Failed to create account: " +
                        creatAccResType.statusInfo.reason);
                }
                else
                {
                    Console.WriteLine("Success!");
                }
            }
        }
    }
```

11. Build and run the application.

FlexNet Authentication Service

The FlexNet Authentication service manages the authentication of a FlexNet Operations user by processing a token passed from another application that performs user authentication, such as a company portal. This section describes the FlexNet Authentication interface.

FlexnetAuthentication Interface presents the XML file that defines the FlexnetAuthentication interface. The interface is used to generate language-specific client implementations that can communicate with the FlexNet Authentication service to manage token authentication of users.

FlexnetAuthentication Interface

The endpoint to connect to the FlexNet Authentication service depends on the Web Service version you are using:

- **Default endpoint**—`https://<host:port>/flexnet/services/FlexnetAuthentication`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/FlexnetAuthentication`

`FlexnetAuthentication.wsdl` defines the FlexnetAuthentication interface and its custom data types, service requests, and service responses.

Device Management Service

The Device Management service enables management of hardware devices. This section describes the ManageDeviceService interface.

Device Management Interface presents the XML files that define the ManageDeviceService interface. The interface is used to generate language-specific client implementations that can communicate with the ManageDeviceService service to manage accounts.

ManageDeviceService Interface

The endpoint to connect to the Device Management service depends on the Web Service version you are using:

- **Default endpoint**—`https://<host:port>/flexnet/services/ManageDeviceService`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/ManageDeviceService`

`ManageDeviceService.wsdl` defines the ManageDeviceService interface and references `ManageDevice.xsd` that defines Device Management service custom data types, service requests, and service responses.

Download Packaging Service

The Download Packaging Service is an add-on to the FlexNet Operations entitlement management system. Its primary components are Files and Download Packages (a collection of Files). Download Packages refer to additional optional attributes of End User License Agreements and Producers which are also defined in this service. Products and Product Lines, which are defined in the Product Service, are also used by the Download Packaging Service.

DownloadPackagingService Interface

The endpoint to connect to the Download Packaging Service depends on the Web Service version you are using:

```
https://<siteID>-fno.flexnetoperations.com/flexnet/services/<version>/DownloadPackagingService
```

Replace <siteID> with your organization’s site ID (supplied by Revenera), and <version> with the desired version. For information about versions of the Download Packaging Service, see Web Service Version History.

`DownloadPackagingService.wsdl` defines the DownloadPackagingService interface.

Version ▪ *As of the FlexNet Operations 2016 R2 release, DownloadPackagingService replaces EsdWebService.*

Usage Service

The Usage Service is a SOAP-based web service that enables producers to retrieve the end-user’s product line, meter, unit of measure, actual use, entitled quantity, overage since statement, overage since reset, percentage, interval/reset, and time period.

UsageService Interface

The endpoint to connect to the UsageService service depends on the Web Service version you are using:

- **Default endpoint**—`https://<host:port>/flexnet/services/UsageService`
- **Version-specific endpoint (for v1)**—`https://<host:port>/flexnet/services/v1/UsageService`

`UsageService.wsdl` defines the UsageService interface.

4

SOAP Web Services Reference

The Using SOAP Web Services section of this guide provides details about integrating your organization’s systems with FlexNet Operations via the SOAP-based Web Service operations, including where to find WSDLs for those services and how to set up a Web Service client.

However, for complete reference information on the individual FlexNet Operations SOAP-based APIs, see the *FlexNet Operations SOAP Web Services Guide*, available from the Revenera Product Documentation site.

5

REST Web Services

FlexNet Operations includes REST services that producers can use to do the following:

- Download data extract snapshots and manage their data extract job output
- Provide communication with end-user applications.

These REST services are described in the following topics:

Table 5-1 ▪ Summary of RESTful services provided by FlexNet Operations

Topic	Description
Data Extract RESTful Services	Details REST API services associated with job-related back office integration
Application RESTful Services	Details REST API services associated with end-user application communication

Data Extract RESTful Services

Data extract REST APIs enable a user to query any jobs that have run for their tenant. The response provides the status for any jobs that are pending, and timestamps for when a job started and ended. The user can also call a service to retrieve an artifact created by the job. For example, the job artifact is often a comma-separated value (CSV) file for data extract jobs. If a job produces two (or more) CSV files, the artifact would be a .zip file that contained all the CSV files.

Important ▪ *Data extract job outputs must be downloaded within a 7 day period from creation. Outputs older than 7 days are automatically purged.*

See the following sections for details about the data extract jobs:

- Using Data Extract REST Endpoints

- Fulfillment Data Extract
- Device Data Extract
- Served Device Data Extract
- User Data Extract
- Account Data Extract
- Usage Data Extract
- Entitlement Data Extract
- Product Packaging Data Extract

Tip ▪ *The data extract jobs do not run by default. Producers can configure one or more of the data extract jobs to run in the Producer Portal (**Administer** > **Configure Alerts/Jobs**). For more information about managing data extract jobs and other alerts, see Administering Alerts in the FlexNet Operations User Guide.*

Using Data Extract REST Endpoints

The following table shows examples of how to view jobs that have run and get the output from those jobs.

Endpoints for data extract job information are available on your FlexNet Operations host:

```
https://<host:port>.flexnet/operations/extracts
```

where *host* is the hostname and *port* is the port number of your FlexNet Operations host.

(The FlexNet Operations host prompts you for your Producer Portal username and password.)

Service URL	Method	Example Response

`.../operations/extracts`	`GET`	Retrieves a list of all jobs that have run.
		
		`/operations/extracts` returns a response like the following:
		
		`[{"id":2087,"jobName":"FULFILLMENT","status":"C","startTime":`
		`"2016-12-12T20:31:54.473Z", "endTime":"2016-12-12T20:31:55.543Z",`
		`"artifactAvailable":false,"filesize":null,"checksum":null},`
		
		`{"id":2088,"jobName":"DEVICE","status":"C","startTime":`
		`"2016-12-09T23:31:54.450Z", "endTime":"2016-12-09T23:31:54.580Z",`
		`"artifactAvailable":true,"filesize":"244","checksum":"B92B84053BA5`
		`EAB1EA78B515793411C3D1EEA438F445433B821B6AB118F3131A"},`
		
		`{"id":2089,"jobName":"USAGE","status":"P","startTime":`
		`"2016-12-09T23:29:39.197Z","endTime":"2016-12-09T23:29:39.337Z",`
		`"artifactAvailable":false,"filesize":null,"checksum":null},`
		
		`{"id":2090,"jobName":"FULFILLMENT","status":"C","startTime":`
		`"2016-12-09T23:28:39.193Z","endTime":"2016-12-09T23:28:39.307Z",`
		`"artifactAvailable":true,"filesize":"244","checksum":"B92B84053BA5`
		`EAB1EA78B515793411C3D1EEA438F445433B821B6AB118F3131A"},`
		
		Optional Parameters
		
		Filter the job entries, if necessary, in the response using optional parameters:
		
		● `name` values include ACCOUNT, DEVICE, ENTITLEMENT, FULFILLMENT, PRODUCTS, SERVEDDEVICE, USAGE, and USER. These values limit the job entries to the entity specified.

Service URL	Method	Example Response
`.../operations/extracts` (continued)	`GET`	● `beginDate` values use the format YYYY-MM-DD (FlexNet Operations server time). ● `endDate` values use the format YYYY-MM-DD (FlexNet Operations server time). ● `status` values can be C for “completed,” F for “failed,” or P for “pending.” ● `artifactAvailable` values can be true or false. If false, no artifact (.csv or .zip file) is available. ● `filesize` (size in bytes) and `checksum` (using the SHA-256 algorithm) are available when two conditions are satisfied: ● `status = C` ● `artifactavailable = true` If both conditions are not satisfied, these fields will be null. Optional parameters can be combined in the request. For example, `https://`*`host:port`*`.flexnet/operations/extracts?name=DEVICE&status=C` returns entries for all completed Device Data Extract jobs.
`.../operations/extracts /{jobid}`	`GET`	Retrieves an entry about the job that matches the job ID. `/operations/extracts/2088` returns `{"id":2088,"jobName":"DEVICE","status":"C","startTime": "2016-12-09T23:31:54.450Z", "endTime":"2016-12-09T23:31:54.580Z", "artifactAvailable":true,"filesize":"244","checksum":"B92B84053BA5 EAB1EA78B515793411C3D1EEA438F445433B821B6AB118F3131A"}`
`.../operations/extracts /{jobid}/file`	`GET`	Retrieves the output of the job that matches the specified job ID. This method returns the contents of the artifact file associated with the job, as a binary stream, if an artifact file location is provided. See entity-specific data extract descriptions, below, for information about the contents of a job’s artifact file. For best performance, consider deleting a job after successfully downloading the job’s artifact file.
`.../operations/extracts /delete/{jobid}`	`POST`	Deletes the job that matches the specified job ID as well as its corresponding artifact file, if any, from its given location.

Authorizing Calls to Data Extract RESTful Services

The Data Extract REST API requires a user to authenticate using an access token.

You can generate an access token in the following ways:

- Using the REST web service **access-token-controller**. This REST web service supports creating, reading, updating, searching, rotating, and deleting tokens, as well as using the tokens to call the APIs. For more information, see the chapter Token-Based Authentication.
- Using the **Manage Access Tokens** page (available from the **Accounts & Users** menu) in the Producer Portal or End-User Portal. For more information, see the section Managing Access Tokens in the *FlexNet Operations User Guide* or Managing Access Tokens in the FlexNet Operations End-User Portal Help. Both documents are available from the Revenera Documentation site.

Fulfillment Data Extract

The first time the Fulfillments Data Extract job runs it creates a snapshot of fulfillment events from the day prior to the day the job runs. (Fulfillments created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job creates a comma-separated value (CSV) file. The job output is a snapshot of fulfillments created or updated; it does not include fulfillment history data.

The first line of the job output contains the header information for the table described by the CSV. If no fulfillments were created or updated in the job period, the CSV output includes only the header row content.

Note ▪ *The date/time format of the data extract is controlled by the* ***Use ISO 8601 time format in extracts*** *configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: Fulfillment Data Extract Job

```
FulfillmentID, Status, ParentFulfillmentID, ActivationID, FulfillmentDate, FulfillCount, OverdraftCount,
ExpirationDate, ShipToEmail, CreatedBy, LicenseModel, FulfillmentSource, LifecycleAction, HostEntityID,
HostUserId, IsServer, Redundant, Trusted, IsNodelocked, IsCustom,

FID_d43cbaf4_3b97_43bf_be75_be9e9ec52503, bo.constants.states.active, null, 709b-434b-6312-4c30-a3f4-
bfdc-c5a9-e329, 2016-10-10 14:35:11.313,1,0, null, my.enduser@mycompany.com, ADMNadmin, Floating
Counted, bo.constants.fulfillmentSource.online, null, ANY, null, true, false, false, false, false,

FID_02bafdb9_835e_44c4_a95a_ef37ed846f00, bo.constants.states.active, null, 709b-434b-6312-4c30-a3f4-
bfdc-c5a9-e329, 2016-10-10 14:35:12.377,1,0, null, my.enduser@mycompany.com, ADMNadmin, Floating
Counted, bo.constants.fulfillmentSource.online, null, ANY, null, true, false, false, false, false,

FID_e0e1ccb4_e62d_408a_a9a1_f1ef0343e132, bo.constants.states.active, null, 709b-434b-6312-4c30-a3f4-
bfdc-c5a9-e329, 2016-10-10 14:35:14.457,1,0, null, my.enduser@mycompany.com, ADMNadmin, Floating
Counted, bo.constants.fulfillmentSource.online, null, ANY, null, true, false, false, false, false,

...
```

Notice that, for fulfillments that have multiple host IDs, FlexNet Operations structures the CSV output so that a new row appears for each additional host ID on a given device. On these extra rows, the fulfillment ID is repeated, the fulfillment details between the fulfillment ID and the additional host ID are left empty, and the additional host ID follows. (See "Example for Multiple Host IDs on a Fulfillment," below.)

Example for Multiple Host IDs on a Fulfillment

```
FulfillmentID, ..., HostID, ...
FID01, ..., Host101, ...,
FID02, ..., Host201, ...
FID02, ..., Host202, ...
FID03, ..., Host301, ...
...
```

Above, FID02 has two host IDs mapped to it: Host201 and Host202. FID01 and FID03 both have only one host ID mapped. In an actual output file, the row for FID02 and Host201 would show the status, the parent fulfillment ID, the activation ID (if any), and so forth. Extra rows for FID02 would not show those values.

Device Data Extract

The first time the Device Data Extract job runs it creates a snapshot of device events (including FlexNet Embedded devices and license servers) from the day prior to the day the job runs. (Device and license servers created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job creates a comma-separated value (CSV) file. The job output is a snapshot of devices and license servers created or updated; it does not include device history data.

The first line of the job output contains the header information for the table described by the CSV. If no devices were created or updated in the job period, the CSV output includes only the header row content.

Sample Output: Device Data Extract Job

```
DeviceID, Type, IDType, BackupDeviceID, Alias, DeviceSeries, PublisherName, DeviceAccountID,
DeviceAccountName, Status, DeviceUserEmail, BaseProduct, SiteName, Notes, ActivationID, LicenseOwnerID,
LicenseOwnerName, ProductName, ProductVersion, QtyOnDevice, LicenseStatus, LicenseModel1, LicenseModel2,
LicenseModel3, IsPermanent, ExpirationDate

Device-1, CLIENT, STRING, , , FLX_CLIENT_SERIES, fnedemo, O1, O1, ACTIVE, , , , CLS-SITE, ba7f-6048-
68a5-4630-b8b1-502a-8548-a115, O1, O1, 3rdLevelUpgradeProd, 3.0, 2, LICENSE_GENERATED, Embedded Counted,
, , NO, Sat Nov 05 00:00:00 PDT 2016,

Device-2, CLIENT, STRING, , , FLX_CLIENT_SERIES, fnedemo, O1, O1, ACTIVE, , FNP_FNE, , , ba7f-6048-68a5-
4630-b8b1-502a-8548-a115, O1, O1, 3rdLevelUpgradeProd, 3.0, 12, LICENSE_GENERATED, Embedded Counted, , ,
NO, Sat Nov 05 00:00:00 PDT 2016,

Device-2, ,,,,,,,,,,,,, 8793-2031-9c1e-4e4a-a5a8-4ff4-8d91-3d70, O1, O1, FNP_FNE, 3.0, 14,
LICENSE_GENERATED, Embedded Uncounted, , , NO, Sat Dec 03 00:00:00 PST 2016,

Device-3, CLIENT, STRING, , , FLX_CLIENT_SERIES, fnedemo, O1, O1, ACTIVE, , , , , ba7f-6048-68a5-4630-
b8b1-502a-8548-a115, O1, O1, 3rdLevelUpgradeProd, 3.0, 3, LICENSE_GENERATED, Embedded Counted, , , NO,
Sat Nov 05 00:00:00 PDT 2016,

...
```

Notice that, for license servers or devices that have multiple activation IDs, FlexNet Operations structures the CSV output so that a new row appears for each additional activation ID on a given device. On these extra rows, the device ID of the device or license server is repeated, the device details between the device ID and the additional activation ID are left empty, and the additional activation ID (as well as information specific to that activation ID) follows. (See “Example for Multiple Activation IDs on a Device,” below.)

Example for Multiple Activation IDs on a Device

```
DeviceID, ..., ActivationID, ...
Device01, ..., Activation101, ...,
Device02, ..., Activation201, ...
Device02, ..., Activation202, ...
Device03, ..., Activation301, ...
...
```

Above, Device02 has two activation IDs mapped to it: Activation201 and Activation202. Device01 and Device03 both have only one activation ID mapped. In an actual output file, the row for Device02 and Activation201 would show the type, the type ID, the alias (if any), and so forth. Extra rows for Device02 would not show those values.

Also notice that, for license servers or devices with activation IDs that have more than one product, FlexNet Operations creates an extra row for each additional product. This follows a very similar pattern to the one used to generate extra rows for multiple activation IDs. In the multiple-products case, both the device ID and the activation ID are repeated, and then additional product name (as well as different information specific to that product) follows. In these extra rows, the device details between the device ID and the activation ID, and the license owner details between the activation ID and the product name, are left empty. (See “Example for Multiple Products in an Activation ID,” below.)

Example for Multiple Products in an Activation ID

```
DeviceID, ..., ActivationID, ..., ProductName, ...
Device01, ..., Activation101, ..., Product1A, ...
...
Device04, ..., Activation401, ..., Product4A, ...
Device04, ..., Activation401, ..., Product4B, ...
Device04, ..., Activation401, ..., Product4C, ...
Device05, ..., Activation501, ..., Product5A, ...
...
```

Above, Device04 has one activation ID mapped to it: Activation401. Activation401 has three products: Product4A, Product4B, and Product4C. Each additional product after Product4A causes an extra row to be generated for Device004.

Extra rows are generated for both activation IDs and products. In the case of multiple activation IDs, the device ID is repeated. In the case of multiple products, the device ID and the activation ID are repeated.

Served Device Data Extract

The first time the Served Device Data Extract job runs it creates a snapshot of served device events from the day prior to the day the job runs. (Served devices created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job creates a comma-separated value (CSV) file. The job output is a snapshot of served devices created or updated; it does not include device history data.

The first line of the job output contains the header information for the table described by the CSV. If no served devices were created or updated in the job period, the CSV output includes only the header row content.

Note *▪ The date/time format of the data extract is controlled by the* ***Use ISO 8601 time format in extracts*** *configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: Served Device Data Extract Job

```
ServedClientHostID, ServerHostID, Alias, SiteName, Notes, DeviceUser, Status, AccountID, AccountName,
DeviceAccountName, LastSyncTime, FeatureName, FeatureVersion, Quantity, Overage, ExpirationDate

clsClient1, 6FK36Q3496KS, ServedDevice, Site-Server-Cloud, CITE_CLS_DEC9,,, Customer, Customer,
Customer, 2016-11-08T19:29:11.900Z, WSfeature3, 1.0, 1, , 1970-01-18T02:53:58.151Z

clsClient2, 6FK36Q3496KS, ,,,,, Customer, Customer, Customer, 2016-11-08T19:29:24.885Z, WSfeature3, 1.0,
1, , 1970-01-18T02:53:58.164Z

clsClient3, 6FK36Q3496KS, ,,,,, Customer, Customer, Customer, 2016-11-08T19:29:31.441Z, WSfeature3, 1.0,
1, , 1970-01-18T02:53:58.171Z

clsClient4, 6FK36Q3496KS, ,,, User_clsClient4, , Customer, Customer, Customer, 2016-11-09T00:25:33.107Z,
WSfeature3, 1.0, 1, , 1970-01-18T02:53:58.178Z

clsClient4, ,,,,,,,,,, WSfeature2, 1.0, 1, , 1970-01-18T02:54:09.994Z

clsClient4, ,,,,,,,,,, WSfeature3, 1.0, 0, , 1970-01-18T02:44:05.194Z

clsClient4, ,,,,,,,,,, WSfeature2, 1.0, 0, , 1970-01-18T02:44:11.133Z

clsClient4, ,,,,,,,,,, WSfeature3, 1.0, 1, , 1970-01-18T02:54:15.933Z

...
```

Note *▪ For served devices that have multiple features, FlexNet Operations structures the CSV output so that a new row appears for each additional feature on a given served device. On these extra rows, the device ID and the feature details are repeated.*

User Data Extract

The first time the User Data Extract job runs it creates a snapshot of user events from the day prior to the day the job runs. (User accounts created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job creates a comma-separated value (CSV) file. The job output is a snapshot of user accounts created or updated; it does not include historical data.

The first line of the job output contains the header information for the table described by the CSV. If no users were created or updated in the job period, the CSV output includes only the header row content. When a user is assigned multiple roles in a given account, the output lists all the user's assigned roles and separates roles with a pipe character (|). Custom attributes, if any, are displayed in columns following the Type column.

The expiry date will display in the template if the **Enable User Expiration** is checked in the system configuration.

***Note** ▪ The date/time format of the data extract is controlled by the **Use ISO 8601 time format in extracts** configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: User Data Extract Job

```
FirstName, LastName, EmailAddress, UserName, Role, Status, AccountID, AccountName, SharedLogin,
PortalLogin, Street, City, State, ZipCode, Country, TimeZone, Locale, PhoneNumber, FaxNumber,
ReceiveEmail, RecieveExpiringEmails, SelfRegisteredUser, CreateDate, LastLoginDate, Type, ExpiryDate,
CA[USER|UserType], CA[USER|SSN]

John, Smith, user-a@flexerasoftware.com, user-a@flexerasoftware.com, Portal User Role|Portal Admin User
Role, Yes, 0814Account1, 0814Acct1, No, Yes, ,,,, US, GMT-8.0DST60, en_US, ,, No, No, No, ,,
CUSTOMER,12/22/2018 0:00,,

Chandra, Smith, user1_extract@gmail.com, user1_extract@gmail.com, Portal User Role|Portal User Role,
Yes, Bravo, bravo_subacct, Yes, Yes, 101 Metro drive, San Jose, CA, 95123, US, GMT-8.0DST60, en_US, 408-
0000, 408-909-0000, Yes, No, ,,, CUSTOMER, Customer, , 123

,,, user1_extract@gmail.com, Portal User Role, , Charlie's Chocolates, Charlie's Chocolates,
,,,,,,,,,,,,,,,,,,,

,,, user1_extract@gmail.com, Portal Admin User Role, , Acme, Acme Software Corporation,
,,,,,,,,,,,,,,,,,,,

Samuel, cust, user-c@flexerasoftware.com, user-c@flexerasoftware.com, Portal Admin User Role, Yes,
Aberdeen Golf, Aberdeen Golf, No, Yes, ,,,, US, GMT-8.0DST60, en_US, ,, No, No, No, ,, CUSTOMER,,,

...
```

Notice that, for users that belong to multiple accounts, FlexNet Operations structures the CSV output so that a new row appears for each additional account. On these extra rows, the UserName of the user is repeated. along with that user's assigned roles in that account and the additional account name. Other details are left empty. (See "Example for Users Linked to Multiple Accounts," below.)

Example for Users Linked to Multiple Accounts

```
FirstName, LastName, EmailAddress, UserName, Role, Status, AccountID, AccountName, ...

Manuel, Enblach, menblach@MIB.com, menblach@MIB.com, Portal User Role, Yes, MIB, MIB, ...

Theresa, Cullen, tcullen@delos.com, tcullen@delos.com, Portal Admin User Role, No, Delos, Delos, ...

Robert, Ford, rford@delos.com, rford@delos.com, Portal Admin User Role, Yes, Delos, Delos, ...

,,, rford@delos.com, Portal Admin User Role|Portal User Role,, Sweetwater, Sweetwater, ...

,,, rford@delos.com, Portal Admin User Role|Portal User Role,, Confederatos, Confederatos, ...

Elsie, Hughes, ehughes@delos.com, ehughes@delos.com, Portal Admin User Role, Yes, Delos, Delos, ...

R, Mistice, armistice@westwrld.com, armistice@westwrld.com, Portal User Role, Yes, Sweetwater,
Sweetwater, ...

...
```

Above, rford@delos.com is linked to three accounts: Delos, Sweetwater, and Confederatos. The other users are each linked to only one account. In the accounts, Sweetwater and Confederatos, rford@delos.com has both portal administrator and portal user roles.

Account Data Extract

The first time the Account Data Extract job runs it creates a snapshot of account events from the day prior to the day the job runs. (Accounts created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job creates a comma-separated value (CSV) file. The job output is a snapshot of accounts created or updated; it does not include historical account data.

The first line of the job output contains the header information for the table described by the CSV. If no accounts were created or updated in the job period, the CSV output includes only the header row content. If there are custom attributes, they appear in the columns following the AccountDescription column. Although the extract does not support hierarchies, it does list the parent accounts if applicable. If more than one parent account exists, they are separated by a pipe (|).

Note ▪ *The date/time format of the data extract is controlled by the* ***Use ISO 8601 time format in extracts*** *configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: Account Data Extract Job

```
AccountID,AccountName,AccountType,ParentAccount,Address1,Address2,City,State,PostalCode,Country,
Visible,Region,AccountDescription,CA[ACCT|CUSTOM_ACCT2222],CA[ACCT|CUSTOM3],CA[ACCT|CUSTOM_ACCT]

MYACCT,My Account,PUBLISHER,,aaaaa,dddd,ccc,ssss,333,US,true,dssd,dddd,,,231

PORTAL_ACCT_UNIT,Portal Account,CUSTOMER,,,,,,,,true,,,,,

UNKNOWN_ACCT_UNIT,Information Not Available,UNKNOWN,,,,,,,,true,,,,,

TEST1,TEST2,CUSTOMER,,add1,add2,cc,sss,6666,US,true,dddd,descrip,789,AAA,197

TEST1_SUB1,TEST1_SUB1,CUSTOMER,TEST1,,,,,,US,true,,,,,

TEST1_PARTNER1,TEST1_PARTNER1,CHANNEL_PARTNER,,,,,,,US,true,,,,,

TEST1_PARTNER1_SUB1,TEST1_PARTNER1_SUB1,CHANNEL_PARTNER,TEST1_PARTNER1,,,,,,US,true,,,,,

MYACCT_SUB1,MYACCTSUB1,PUBLISHER,MYACCT,,,,,,US,true,,,,,

user-n@flexerasoftware.com-FLEX1000,FLEX1000,SELF_REGISTERED,,,,,,,,true,,,,,

TEST1_SUB1_SUB2,TEST1_SUB1_SUB2,CUSTOMER,TEST1_SUB1,TEST1_SUB1_SUB2,TEST1_SUB1_SUB2,

TEST1_SUB1_SUB2,TEST1_SUB1_SUB2,,US,true,TEST1_SUB1_SUB2,TEST1_SUB1_SUB2,,,23

TWOPARENTS,TWOPARENTS,CUSTOMER,TEST1|TEST1_SUB1,TWOPARENTS,TWOPARENTS,TWOPARENTS,TWOPARENTS,,US,

true,TWOPARENTS,TWOPARENTS,,,12

...
```

Usage Data Extract

The first time the Usage Data Extract job runs it creates a snapshot of usage events from the day prior to the day the job runs. (Usage events from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job creates a comma-separated value (CSV) file. The first line of the job output contains the header information for the table described by the CSV. If no usage events occurred in the job period, the CSV output includes only the header row content.

***Note** ▪ The date/time format of the data extract is controlled by the **Use ISO 8601 time format in extracts** configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: Usage Data Extract Job

```
Account, Product, Feature, Transaction Time, Received Time, Count, Acquisition ID, Consumer ID, Device
ID, Server ID

"Usage Account", "Uncategorized Products", "Usage_Feature", 2016-03-11T07:41:38.-0600,2016-03-
11T07:42:11.-0600,1,,,"TestHost","7KV4HEY2XFWU"

"Usage Account", "Uncategorized Products", "Usage_Feature", 2016-03-11T07:41:40.-0600,2016-03-
11T07:42:11.-0600,1,,,"TestHost","7KV4HEY2XFWU"

"Usage Account", "Uncategorized Products", "Usage_Feature", 2016-03-11T07:41:41.-0600,2016-03-
11T07:42:11.-0600,1,,,"TestHost","7KV4HEY2XFWU"

...
```

The structure of this job output file matches the structure used for downloads of usage summaries from the Producer Portal.

Entitlement Data Extract

The first time the Entitlements Data Extract job runs it creates a snapshot of fulfillment events from the day prior to the day the job runs. (Entitlements created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job results in the extract information getting downloaded as a zip file (not as a CSV file). This is different from the other extracts. The zip file contains two CSV files – one for entitlement line items and the other for maintenance line items. As an example, the name of the zip file would be Entitlements-2017-07-07-02-27-18.zip and inside this the name of the CSV's would be EntitlementsExtract-2017-07-05-03-07-21.csv (entitlement line items) and MaintenanceLine-2017-07-05-03-07-21.csv (for maintenance line items). The naming format in these files is "YYYY-MM-DD-HH-MM-SS". The job output is a snapshot of fulfillments created or updated; it does not include entitlement history data. The first line of the job output contains the header information for the table described by the CSV. If no entitlements were created or updated in the job period, the CSV output includes only the header row content.

***Note** ▪ The date/time format of the data extract is controlled by the **Use ISO 8601 time format in extracts** configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: Entitlement Data Extract Job - Entitlement Line Items

```
EntitlementID,EntitlementState,EntitlementDescription,AllowPortalLogin,ShipToEmail,Address,End
Customer|Account,End Customer|AccountName,End Customer|UserName,End Customer|UserFirstName,End
Customer|UserLastName,End Customer|UserEmail,End Customer|UserPhone,Partner Tier 1|Account,Partner Tier
1|AccountName,Partner Tier 1|UserName,Partner Tier 1|UserFirstName,Partner Tier 1|UserLastName,
Partner Tier 1|UserEmail,Partner Tier 1|UserPhone,CurrentOwner,ActivationId,ActivationState,
LineItemQuantity,OrderLineNumber,OrderId,StartDateType,StartDate,IsActivationPermanent,EndDate,EndDurat
ionUnits,EndDuration,VersionDateType,VersionDate,VersionDurationUnits,VersionDuration,OrderType,OrderPa
rent,LineItemDescription,ProductName,ProductVersion,ProductType,ProductCount,PartNumber,LicenseModel,
LM[Embedded Counted|NOTICE],LM[Embedded Counted|SERIAL_NUMBER],LM[Embedded Counted|VENDOR_STRING],
LM[Floating Uncounted|NOTICE],LM[Floating Uncounted|SERIAL_NUMBER],LM[Nodelocked Uncounted|NOTICE],
LM[Nodelocked Uncounted|SERIAL_NUMBER],LM[Embedded Uncounted|NOTICE],LM[Embedded
Uncounted|SERIAL_NUMBER],LM[Embedded Uncounted|VENDOR_STRING],LM[Nodelocked
Counted|NOTICE],LM[Nodelocked Counted|SERIAL_NUMBER],LM[Floating Counted|NOTICE],LM[Floating
Counted|LM_TESTING],LM[Floating Counted|SERIAL_NUMBER]
```

```
1ad4-60c4-be37-4df6-a456-584c-51f9-0bd9,Deployed,My Example Entitlement,Yes,test@customer1.org,"100,
California Avenue, USA",Acct1,Acme,endUser1,John,Smith,john.smith@earthday.com,111-222-
3334,Acct2,Acme2,partnerUser1,Jack,Jill,jack.jill@nurseryrhymes.com,999-999-
9998,bo.constants.partnertiernames.endcustomer,enterpriseAct,Deployed,100,,,DefineNow,30:00.0,TRUE,,,,,
,,,,1ad4-60c4-be37-4df6-a456-584c-51f9-0bd9,,Enterprise,1,LicensedProduct,1,,Enterprise
Activation,,,,,,,,,,,,,,
```

The sections **Account** to **UserPhone** will be repeated for each reseller/tier. For example, if the entitlement had “End Customer” and “Partner Tier 1” (2 tiers only), then that particular section would look like

- End Customer|Account
- End Customer|UserName
- End Customer|UserFirstName
- End Customer|UserLastName
- End Customer|UserEmail
- End Customer|UserPhone
- Partner Tier 1|Account
- Partner Tier 1|AccountName
- Partner Tier 1|UserName
- Partner Tier 1|UserFirstName
- Partner Tier 1|UserLastName
- Partner Tier 1|UserEmail
- Partner Tier 1|UserPhone

Notice that, for entitlements that have multiple line items (or activation IDs), FlexNet Operations structures the CSV output so that a new row appears for each activation ID. On these extra rows, the entitlement ID is repeated, the entitlement details between the entitlement ID and the activation ID are left empty.

Example of Multiple Activation IDs on an Entitlement

```
EntitlementID, ..., ActivationId, ...
EID01, ..., act101, ...,
EID02, ..., act201, ...,
EID02, ..., act301, ...,
EID03, ..., act401, ...,
...
```

Sample Output: Entitlement Data Extract Job - Maintenance Line Items

```
EntitlementID,EntitlementState,EntitlementDescription,AllowPortalLogin,ShipToEmail,Address,ActivationId
,LineItemLink,ActivationState,ProductName,ProductVersion,PartNumber,OrderLineNumber,OrderId,StartDate,I
sActivationPermanent,EndDate,CA[MAINT_LINE|cA_MNT_LI_B2],CA[MAINT_LINE|CA1M],CA[MAINT_LINE|cA_MNT_LI_B1
],CA[MAINT_LINE|CAm2],CA[MAINT_LINE|BMLICA1],CA[MAINT_LINE|TESTCA_Ent_MNT],CA[MAINT_LINE|DateCA_MLI]
```

```
Nu-a270-0047-487b-651,Deployed,,Yes,,,actmain_01,,Deployed,maintenance19082016,1.1,,8232016,8232016,Mon
Aug 22 17:00:00 PDT 2016,FALSE,Wed Aug 30 05:30:00 IST 2017,1234567,A2,123456,A1M|A2M,TRUE,12345,24-Aug-
17
```

In the maintenance CSV file, the ActivationId refers to the maintenance line item ID and the LineItemLink refers to the linked entitlement line item ID. If there are multiple maintenance items linked to the same line item, a new row appears for each activation ID. On these extra rows, the entitlement ID and the line item link are repeated and the details between the entitlement ID and the activation ID are left empty.

The format for custom attribute columns is: `CA[MAINT_LINE|`*`name`*`]`, where *name* is the name of the custom attribute for a maintenance line item. The custom attribute column can appear as many times as there are custom attributes defined for the maintenance line item.

Example of Multiple Maintenance Items Attached to the Same Line Item

```
EntitlementID, ..., activationID, LineItemLink..
EID01, ..., act101, lil01, ..,
EID01, ..., act201, lil01, ..,
EID01, ..., act301, lil01, ..,
EID01, ..., act401, lil01, ..,
...
```

Product Packaging Data Extract

The first time the Product Packaging Data Extract job runs it creates a snapshot of product events from the day prior to the day the job runs. (Products created or updated from 00:00:00 to 23:59:59, FlexNet Operations server time, on the previous day are captured when the job runs). Subsequently, when this job runs, it creates a snapshot from the end time of the last successful run to the current time.

A successful run of this job results in the extract information getting downloaded as a zip file. The output contains the details of products and their dependent entities (such as maintenance, suites, part numbers, and license models). It also captures features (including their version, state, and quantity) that are associated with the products. The job output is a snapshot of products created or updated; it does not include historical product data.

The name of the zip file would be `Products-<date and time>.zip`. Inside the zip file, the output is grouped into four CSV files which contain the following information:

- **ProductsExtract-<date and time>.csv**—Contains the main product record information and details about dependent entities
- **ProductsLicenseModel-<date and time>.csv**—Contains the license model details that are associated with the products
- **ProductsPartNumber-<date and time>.csv**—Contains details about the part numbers associated with the products
- **ProductsRelation-<date and time>.csv**—Contains details about the relationships between the products and the related products

The naming format in the zip file and the CSV files is “YYYY-MM-DD-HH-MM-SS” (for example, **Products-2019-06-07-16-29-10.zip**).

The first line of the job output contains the header information for the table described by the CSV. If no products were created or updated in the job period, the CSV output includes only the header row content.

***Note** ▪ The date/time format of the data extract is controlled by the **Use ISO 8601 time format in extracts** configuration option. For more information, see Configuring System Settings | General Options in the FlexNet Operations User Guide.*

Sample Output: Product Packaging Data Extract Job - Product Extract

```
ProductName, OrderableType, Description, Version, State, PackageName, PackageVersion,
PackageVersionFormat, LicenseTechnologyName, UsedOnaDevice, UsageModel, LicenseGenerator, FnpVersion,
AllowDownloadObsoleteFrInPortal, AllowDownloadObsoleteFrInAdmin, StartDate, EndDate, Product Line,
AllowUpgrades, AllowUpsells, AllowRenewals, UpgradeEmailTemplate, Feature|Bundle Name, FeatureType,
FeatureVersion, FeatureCount, FeatureDescription, FeatureState, SuiteProductName, SuiteProductVersion,
SuiteProductState, SuiteProductQty, CA[PROD|Prod_boolean_new], CA[PROD|product_NUMBER],
CA[PROD|Product_boolean], CA[PROD|Product_mal_val], CA[PROD|product_date], CA[PROD|TestProduct],
CA[PROD|Prd], CA[PROD|Product_boolean1]

ExtractMaintenance1556128254153,bo.constants.orderable.type.maintenance,test,1,bo.constants.states.draf
t,,,,,,,,,,,,,Uncategorized Products,FALSE,FALSE,TRUE,,,,,,,,,,,,,,

Suites_extract1,bo.constants.orderable.type.uniform_suite,Suites_extract11,2019/21,bo.constants.states.
draft,Extract121,4564,Fixed,NEW_LIC_TECH,FALSE,,TEST_LIC_GEN_CONF,,TRUE,FALSE,,,UncategorizedProducts,,
,,,,,,,,Product_extract1,1.0,bo.constants.states.deployed,2,FALSE,,FALSE,,,sdgsfg,fsdfg,FALSE

PhotoPrint,bo.constants.orderable.type.licensedProduct,,2,bo.constants.states.deployed,PACKAGE,1234,Fix
ed,FlexNet Licensing,TRUE,None,demo,7.1 | 10.1 | 10.8,FALSE,FALSE,2019-04-26T05:30:00Z,2019-05-
31T05:30:00Z,Uncategorized Products,,,,DEFAULT,F1,FEATURE,2,12,,DEPLOYED,FALSE,142,TRUE,,11-Apr-
19,,TEST12,FALSE
...
```

When a product supports multiple FNP versions, the output lists all supported FNP versions and separates them with a pipe character (|). The `FeatureType` column identifies whether the item is a feature or a bundle; possible values are FEATURE and FEATURE BUNDLE. Custom attributes, if any, are displayed in columns following the `FeatureState` column.

Notice that, for products that contain multiple features or feature bundles, FlexNet Operations structures the CSV output so that a new row appears for each additional feature or feature bundle. On these extra rows, the product name of the product is repeated. Other details are left empty. (See Example for Products Containing Multiple Features or Feature Bundles, below.)

Example for Products Containing Multiple Features or Feature Bundles

```
ProductName, OrderableType, Description, Version, State, PackageName, PackageVersion,
PackageVersionFormat, LicenseTechnologyName, UsedOnaDevice, UsageModel, LicenseGenerator, FnpVersion,
AllowDownloadObsoleteFrInPortal, AllowDownloadObsoleteFrInAdmin, StartDate, EndDate, Product Line,
AllowUpgrades, AllowUpsells, AllowRenewals, UpgradeEmailTemplate, Feature|Bundle Name, FeatureType,
FeatureVersion, FeatureCount, FeatureDescription, FeatureState, CA[PROD|Prod_boolean_new],
CA[PROD|product_NUMBER], CA[PROD|Product_boolean], CA[PROD|Product_mal_val], CA[PROD|product_date],
CA[PROD|TestProduct], CA[PROD|Prd], CA[PROD|Product_boolean1]

PhotoPrint,bo.constants.orderable.type.licensedProduct,,2,bo.constants.states.deployed,PACKAGE,1234,Fix
ed,FlexNet Licensing,TRUE,None,demo,7.1 | 10.1 | 10.8,FALSE,FALSE,2019-04-26T05:30:00Z,2019-05-
```

```
31T05:30:00Z,Uncategorized Products,,,,DEFAULT,F1,FEATURE,2,12,,DEPLOYED,FALSE,142,TRUE,,11-Apr-
19,,TEST12,FALSE

PhotoPrint,,,,,,,,,,,,,,,,,,,,,,FB1,FEATURE BUNDLE,,4,,DEPLOYED,,,,,,,,

...
```

Above, the product PhotoPrint contains a feature called F1 and a feature bundle called FB1.

Sample Output: Product Packaging Data Extract Job - Product License Model

```
ProductName,ProductVersion,LicenseModel,DefaultTransactionKey,VirtualTransactionKey
PhotoPrint,v1,Enterprise Activation,,Demo1
PhotoPrint,v1,Lic Model_Nov21_1,,
Prod_new,v4.2,Embedded Counted,,
...
```

In this example, the product PhotoPrint is linked to two license models (`Enterprise Activation` and `Lic Model_Nov21`). FlexNet Operations structures the CSV output so that a new row appears for each additional license model. On these extra rows, the product name of the product is repeated. Other details are left empty.

The columns `DefaultTransactionKey` and `VirtualTransactionKey` are only applicable to trusted license models.

The column `ProductVersion` is only included if the configuration option **Show product version in extract file** (under **System** > **Configure** > **FlexNet Operations** > **Embedded Device Settings**) is selected.

Sample Output: Product Packaging Data Extract Job - Product Part Number

```
ProductName,ProductVersion,PartNumber,PartNumberDescription,AvailableForTrial,RelativeLicenseModel
PhotoPrint,2,PN_NT,,FALSE,FlexNet Licensing
PhotoPrint,2,PN_T,,TRUE,Enterprise Activation
...
```

For products that are mapped to more than one part number, FlexNet Operations structures the CSV output so that a new row appears for each additional part number. On these extra rows, the product name of the product is repeated. Other details are left empty.

The column `RelativeLicenseModel` indicates the license model that is mapped to the respective part number (optional).

Sample Output: Product Packaging Data Extract Job - Product Relations

```
ProductName,ProductVersion,ProductType,RelationshipType,RelatedProductName,RelatedProductVersion,Relate
dProductType

ExtractMaintenance1556128254153,1,bo.constants.orderable.type.maintenance,bo.constants.orderable.relati
onType.isMaintenance,P1.32_1,2018-21,bo.constants.orderable.type.licensedProduct

ExtractMaintenance1556128254153,1,bo.constants.orderable.type.maintenance,bo.constants.orderable.relati
onType.isMaintenance,ExtractProduct1556128254153,1,bo.constants.orderable.type.licensedProduct

Suites_extract1,2019/21,bo.constants.orderable.type.uniform_suite,bo.constants.orderable.relationType.d
emoOf,Suites_extract1,2019/21,bo.constants.orderable.type.uniform_suite

Suites_extract1,2019/21,bo.constants.orderable.type.uniform_suite,bo.constants.orderable.relationType.P
roductionVersionOf,Suites_extract1,2019/21,bo.constants.orderable.type.uniform_suite

PhotoPrint,2,bo.constants.orderable.type.licensedProduct,bo.constants.orderable.relationType.upgradeTo,
PhotoPrint,2,bo.constants.orderable.type.licensedProduct

PhotoPrint,2,bo.constants.orderable.type.licensedProduct,bo.constants.orderable.relationType.hasMainten
ance,CM,1,bo.constants.orderable.type.maintenance

PhotoPrint,2,bo.constants.orderable.type.licensedProduct,bo.constants.orderable.relationType.upgradeFro
m,PhotoPrint,2,bo.constants.orderable.type.licensedProduct

...
```

For products that are mapped to several other products, FlexNet Operations structures the CSV output so that a new row appears for each additional related product. On these extra rows, the product name of the product is repeated. Other columns are populated as appropriate.

Application RESTful Services

Version ▪ *As of 2018 R2, documentation of application RESTful services has been moved to Swagger. Use the following URL to access the information:*

`https://YOUR_FNO_INSTANCE/flexnet/swagger-ui.html`

For example, if your FNO instance was `https://prod2222.mycompany.com`, then the URL to the Swagger documentation would be:

`https://prod2222.mycompany.com/flexnet/swagger-ui.html`

RESTful services information will no longer be maintained in this document.

REST APIs enable an end-user application to retrieve and send product and device information with FlexNet Operations.

API	Description
Entitlement Orders	Returns the activatable line items that match specified criteria
Entitlement Orders Count	Returns the number of activatable line items that match specified criteria.

API	Description
Devices	Returns the query parameters for a specified device (or all devices if no parameters are specified)
Devices Count	Returns the number of matching devices to one or more query parameters or all devices if no parameters specified.
Consumption	Returns the consumption information for a given account ID.

Note ▪ *All date values use the yyyy-MM-dd format.*

REST Web Services Batch Limits

The following batch-size maximum limits apply for requests issued from application RESTful web services:

- All read services have a maximum batch size of 2000.
- All services that result in a write have a batch size maximum of 25 records.

Specifying a batch size higher than the new cap results in an error.

Authorizing Calls to Application RESTful Services

When calling the API, a producer user must authenticate themselves and have the necessary permissions.

The following sections describe ways to provide credentials and note the permissions required for accessing the API:

- Token-Based Authentication—This chapter describes how to authenticate using an access token. This is the recommended authentication method.
- Basic Authentication for Application RESTful Services—This section describes how to provide credentials for basic authentication. This authentication method is considered less secure than token-based authentication.
- Permissions for Application RESTful Services

Basic Authentication for Application RESTful Services

When calling the API, a producer user must authenticate themselves. Authentication information (in Base64) must be included in the header. The following procedure explains how to provide credentials, including an example for each step.

Task ***To provide credentials***

1. Concatenate the user name and password with a single colon.

   Example: If the user name is "alice" and the password is "password", the resulting concatenated string would be `alice:password`

2. Base64 encode the concatenated user name and password.

   Example: Base64 encode the string `alice:password` to obtain "YWxpY2U6cGFzc3dvcmQ="

3. Add the "Authorization" key to the header of the request, with the Base64-encoded result as the value.

   Example: Add the value "Basic YWxpY2UucGFzc3dvcmQ=" to the Authorization header.

Permissions for Application RESTful Services

All REST APIs require the user to be logged into the server and have permissions to execute web services as well as permission for the resource, i.e. View Entitlement, Manage Devices, etc.

Entitlement Orders

The Entitlement Orders service returns the activatable line items that match specified criteria.

***Note** ▪ This operation does not return maintenance line items*

Endpoint

`<HOST>/flexnet/operations/entitlementOrders`

Method

POST

URL Params

None

Data Params

None

Example

```
{"activatableItemSearchCriteria":
    {"parentBulkEntitlementId":
        {"value":"[String to match all or part of the entitlement ID of the parent bulk entitlement
                  of a simple entitlement created when a web register key is redeemed]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "entitlementId":
        {"value":"[String to match all or part of the entitlement ID]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "activationId":
        {"value":"[String to match all or part of the activation ID]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "productName":
```

```
    {"value":"[String to match all or part of the product name]",
    "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
},
"productVersion":
    {"value":"[String to match all or part of the product version]",
    "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
},
"productType":[One of the following: LICENSED_PRODUCT, SUITE, or MAINTENANCE],
"partNumber":
    {"value":"[String to match all or part of the part number]",
    "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
},
"startDate":
    {"value":"[Date value to compare to the start date value]",
    "searchType":"[One of the following: BEFORE, AFTER or ON]",
},
"isPermanent":[Set to true if the entitlement is permanent; false if the entitlement expires],
"expirationDate":
    {"value":"[Date value to compare to the expiration date value]",
    "searchType":"[One of the following: BEFORE, AFTER or ON]",
},
"versionDate":
    {"value":"[Date value to compare to the version date value]",
    "searchType":"[One of the following: BEFORE, AFTER or ON]",
},
"licenseTechnology":
    {"value":"[String to match all or part of the entitlement ID]",
    "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
},
"orderId":
    {"value":"[String to match all or part of the order ID]",
    "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
},
"withNoOrderId":"[Optional. If true, return only activatable items with no order ID. Since this
                 field is intended to return results without the specified attribute, setting
                 this to false will have no effect on query results.]",
"restrictToItemsWithCount":"[Optional. If true, return only activatable items that have a
                            non-zero remaining count. Since this field is intended to return
                            results without the specified attribute, setting this to false
                            will have no effect on query results.]",
"fulfilledAmount":[The number of copies fulfilled]",
"userId":[Set to the user name (login) of a portal user to return the activatable items that
         this user can see if logged in to the portal UI],
"customAttributes":
    {"attribute":[{
        {"attributeName":"[The name of the custom attribute]",
        [Choose one of the following]
        "booleanValue":"[Optional. Depending on the attribute type, set either stringValue or
                        numberValue or booleanValue. Set to true or false if the custom attribute
                        is of type boolean.]",
        "dateValue":
            {"value":"[Optional. Date value to compare to the custom attribute date value]",
            "searchType":"[One of the following: BEFORE, AFTER, ON]"
            },
        "numberValue":
            {"value":"[Optional. An integer number to match part or all of the custom attribute
```

```
                      value.]",
                "searchType":"[One of the following: GREATER_THAN, GREATER_THAN_EQUALS, LESS_THAN,
                              LESS_THAN_EQUALS, EQUALS, or NOT_EQUALS.]"
                },
            "stringValue":
                {"value":"[Optional. A string to match part or all of the custom attribute value]",
                "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
                }
            }]
        },
    "soldTo":
        {"value":"[Optional. A string to match part or all of the account that has the right
                  to fulfill or distribute activation IDs from this entitlement.]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
    "parentBulkEntSoldTo":
        {"value":"[A string to match part or all of the account to which the bulk entitlement
                  was sold]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
    "activatableItemType":[One of the following: LINEITEM or WEBREGKEY],
    "allowPortalLogin":[True if the entitlement ID can be used to log in to the self-service portal;
                       otherwise, false],
    "currentOwnerName":
        {"value":"[String to match all or part of the user name of the user who created the
                  entitlement]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
    },
    "lineItemParentLineItemId":
        {"value":"[A string to match part or all of the line item ID (activation ID) of a
                   parent line item]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
    "createdOnDateTime":
        {"value":"[Date value to compare to the created date and time value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "lastModifiedDateTime":
        {"value":"[Date value to compare to the last modified date and time value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "lineItemAttributes":
        {"attributeName":"[Name of a text-type line item custom attribute by which to search for a
                          line item.]",
        "stringValue":"[Value of the text-type line item custom attribute]",
    },
    "accountId":
        {"value":"[The name of the account]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
        "partnerTier":[Optional. The name of the tier in which the account exists or ANY.
                      If set to a named tier, the format partnerTier is
                      bo.constants.partnertiernames.tier_name. If not specified, the partner
                      tier searched is the End Customer tier.]
    },
    "batchSize":[Required. Maximum number of records to return in the response. e.g. 100],
    "pageNumber":[Defaults to 1. For example: If query returns 100 records and the batch size is
```

```
                 set to 50, enter 2 to see the second page.]
}
```

Response

```
{
    "statusInfo":
    {
        "status": "SUCCESS",
        "reason": null
    },
    "activatableItem":
    [
        {
            "activatableItemType": "LINEITEM",
            "parentBulkEntitlementId": null,
            "entitlementId": "K_Ent2",
            "soldTo": "ACME",
            "shipToEmail": "",
            "shipToAddress": "",
            "entitlementState": "DEPLOYED",
            "activatableItemData":
            {
               "activationId":
               {
                  "id": "K_Act2",
                  "autoGenerate": null
               },
               "description": "",
               "product":
               {
                  "uniqueId": "HID-127905",
                  "primaryKeys":
                  {
                     "name": "K_Prod2",
                     "version": "1.0"
                  }
               },
               "partNumber": null,
               "licenseModel":
               {
                  "uniqueId": "HID-224",
                  "primaryKeys":
                  {
                     "name": "Embedded Counted"
                  }
               },
               "alternateLicenseModel1": null,
               "alternateLicenseModel2": null,
               "licenseModelAttributes":
               {
                  "attribute":
                  [
                     {
                        "attributeName": "NOTICE",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     },
                     {
                        "attributeName": "ENTITLEMENT_TYPE",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     },
                     {
                        "attributeName": "Num_attr2",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     },
                     {
                        "attributeName": "Trial",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     },
                     {
                        "attributeName": "flexnet_lm_attr_text",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     },
                     {
                        "attributeName": "SERIAL_NUMBER",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     },
                     {
                        "attributeName": "VENDOR_STRING",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": null,
                        "integerValue": null,
                        "arrayValue": null
                     }
                  ]
               },
               "policyAttributes":
```

```
{
   "rehostsPolicy": null,
   "returnsPolicy": null,
   "repairsPolicy": null,
   "extraActivationsPolicy": null,
   "cancelLicensePolicy":
   {
      "isCancelLicense": false
   },
      "virtualLicensePolicy":
   {
      "isVirtualLicense": true
   },
      "reinstallPolicy": null,
      "acpiGenerationIdLicensePolicy":
   {
      "useACPIGenerationId": false
   }
   },
      "orderId": null,
      "orderLineNumber": "",
      "numberOfCopies": 100,
      "startDate": 1506495600000,
      "startDateOption": "DEFINE_NOW",
      "isPermanent": false,
      "term":
   {
      "numDuration": 95,
      "durationUnit": "DAYS"
   },
      "expirationDate": 1514707200000,
      "versionDate": null,
      "versionDateAttributes": null,
      "lineItemType": null,
      "entitledProducts":
   {
      "entitledProduct":
      [
         {
            "product":
               {
                  "uniqueId": "HID-127905",
                  "primaryKeys":
               {
                  "name": "K_Prod2",
                  "version": "1.0"
               }
            },
             "quantity": 1
         },
         {
             "product":
             {
                 "uniqueId": "HID-127904",
                 "primaryKeys":
                 {
                     "name": "K_Prod1",
                     "version": "1.0"
                 }
             },
             "quantity": 1
         }
     ]
 },
 "lineItemAttributes":
 {
     "attribute":
     [
         {
             "attributeName": "Boolean1",
             "stringValue": null,
             "dateValue": null,
             "booleanValue": false,
             "integerValue": null,
             "arrayValue": null
         },
         {
             "attributeName": "ENT_LINE_Boolean_B",
             "stringValue": null,
             "dateValue": null,
             "booleanValue": false,
             "integerValue": null,
             "arrayValue": null
         },
         {
             "attributeName": "Showa_Line_bool",
             "stringValue": null,
             "dateValue": null,
             "booleanValue": false,
             "integerValue": null,
             "arrayValue": null
         }
     ]
 },
 "numberOfRemainingCopies": 100,
 "availableExtraActivations": 0,
 "isTrustedType": false,
 "state": "DEPLOYED",
 "licenseTechnology":
 {
     "uniqueId": "HID-34",
     "primaryKeys":
     {
         "name": "FlexNet Licensing"
     }
 },
 "parentLineItem": null,
 "createdOnDateTime": 1506497656047,
 "lastModifiedDateTime": 1506497670777,
 "overdraftMax": 0,
 "remainingOverdraftCount": 0,
 "id": 594475,
    "fnptimeZoneValue": null
},
"channelPartners":
{
    "channelPartner":
    [
        {
            "tierName": "bo.constants.partnertiernames.endcustomer",
            "contact":
            {
                "uniqueId": "HID-32009",
                "userName": "A111@gmail.com",
                "primaryKeys":
                {
                    "firstName": "NK1",
                    "lastName": "11",
                    "emailAddress": "A111@gmail.com",
                    "phoneNumber": " "
                }
            },
            "currentOwner": true,
            "account":
            {
                "uniqueId": "HID-438",
                "primaryKeys":
                {
                    "name": "ACME"
                }
            }
        }
    ]
},
"entitlementAttributes":
{
    "attribute":
    [
        {
            "attributeName": "Boolean1",
            "stringValue": null,
            "dateValue": null,
            "booleanValue": false,
            "integerValue": null,
            "arrayValue": null
        },
        {
            "attributeName": "CA_ENT_Boolean_B",
            "stringValue": null,
            "dateValue": null,
            "booleanValue": false,
            "integerValue": null,
            "arrayValue": null
        },
        {
            "attributeName": "Mandate",
            "stringValue": null,
            "dateValue": null,
                        "booleanValue": false,
                        "integerValue": null,
                        "arrayValue": null
                    },
                    {
                        "attributeName": "my_ent_attr",
                        "stringValue": null,
                        "dateValue": null,
                        "booleanValue": false,
                        "integerValue": null,
                        "arrayValue": null
                    }
                ]
            }
        }
    ]
}
```

Entitlement Orders Count

The Entitlement Orders Count service returns the number of activatable line items that match specified criteria.

Endpoint

<HOST>/flexnet/operations/entitlementOrders/count

Method

POST

URL Params

None

Data Params

None

Example

```
{"queryParams":
    {"parentBulkEntitlementId":
        {"value":"[String to match all or part of the entitlement ID of the parent bulk entitlement
                   of a simple entitlement created when a web register key is redeemed]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "entitlementId":
        {"value":"[String to match all or part of the entitlement ID]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "activationId":
        {"value":"[String to match all or part of the activation ID]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "productName":
        {"value":"[String to match all or part of the product name]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "productVersion":
        {"value":"[String to match all or part of the product version]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "productType":[One of the following: LICENSED_PRODUCT, SUITE, or MAINTENANCE],
    "partNumber":
        {"value":"[String to match all or part of the part number]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
    },
    "startDate":
        {"value":"[Date value to compare to the start date value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "isPermanent":[Set to true if the entitlement is permanent; false if the entitlement expires],
    "expirationDate":
        {"value":"[Date value to compare to the expiration date value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "versionDate":
        {"value":"[Date value to compare to the version date value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "licenseTechnology":
        {"value":"[String to match all or part of the entitlement ID]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
    },
    "orderId":
        {"value":"[String to match all or part of the order ID]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "withNoOrderId":[Optional. If true, return only activatable items with no order ID. Since this
                    field is intended to return results without the specified attribute, setting
                    this to false will have no effect on query results.],
    "restrictToItemsWithCount":[Optional. If true, return only activatable items that have a
                               non-zero remaining count. Since this field is intended to return
                               results without the specified attribute, setting this to false will
                               have no effect on query results.],
    "fulfilledAmount":[The number of copies fulfilled],
    "userId":[Set to the user name (login) of a portal user to return the activatable items that
             this user can see if logged in to the portal UI],
    "customAttributes":
        {"attribute":[{
            {"attributeName":"[The name of the custom attribute]",
            [Choose one of the following]
            "booleanValue":"[Optional. Depending on the attribute type, set either stringValue or
                            numberValue or booleanValue. Set to true or false if the custom attribute
                            is of type boolean.]",
            "dateValue":
                {"value":"[Optional. Date value to compare to the custom attribute date value]",
                "searchType":"[One of the following: BEFORE, AFTER, ON]"
            },
            "numberValue":
                {"value":"[Optional. An integer number to match part or all of the custom attribute
                           value.]",
                "searchType":"[One of the following: GREATER_THAN, GREATER_THAN_EQUALS, LESS_THAN,
                               LESS_THAN_EQUALS, EQUALS, or NOT_EQUALS.]"
            },
            "stringValue":
                {"value":"[Optional. A string to match part or all of the custom attribute value]",
                "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
            }
        }]
    },
    "soldTo":
        {"value":"[Optional. A string to match part or all of the account that has the right to fulfill
                   or distribute activation IDs from this entitlement.]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "parentBulkEntSoldTo":
        {"value":"[A string to match part or all of the account to which the bulk entitlement
                  was sold]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "activatableItemType":[One of the following: LINEITEM or WEBREGKEY],
    "allowPortalLogin":[True if the entitlement ID can be used to log in to the self-service portal;
                        otherwise, false],
    "currentOwnerName":
        {"value":"[String to match all or part of the user name of the user who created the
                  entitlement]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
    },
    "lineItemParentLineItemId":
        {"value":"[A string to match part or all of the line item ID (activation ID) of a
                   parent line item]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
    },
    "createdOnDateTime":
        {"value":"[Date value to compare to the created date and time value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "lastModifiedDateTime":
        {"value":"[Date value to compare to the last modified date and time value]",
        "searchType":"[One of the following: BEFORE, AFTER or ON]",
    },
    "lineItemAttributes":
        {"attributeName":"[Name of a text-type line item custom attribute by which to search for a
                          line item.]",
"stringValue":"[Value of the text-type line item custom attribute]",
    },
    "accountId":
        {"value":"[The name of the account]",
        "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]",
        "partnerTier":[Optional. The name of the tier in which the account exists or ANY.
                      If set to a named tier, the format partnerTier is
                      bo.constants.partnertiernames.tier_name. If not specified, the partner
                      tier searched is the End Customer tier.]
    },
    "restrictToItemsReadyToActivate":[Optional. Restricts the search to only items ready to activate.
```

```
                                        One of the following: True or False.]
}
```

Response

```
{
   "statusInfo":    {
      "status": "SUCCESS",
      "reason": null
   },
   "count": 1869
}
```

Devices

The Devices Query operation takes one or more query parameters to return the matching devices. If no specific query parameters are provided in the `queryParams` element, all devices are returned

***Note** ▪ Some elements apply only to served client devices while others apply only to server devices and client devices. Elements that start with feature are specific to served clients. Elements that start with addOn or preBuilt are specifc to server devices and client devices.*

Query results can be restricted to one or more device types listed in the deviceTypes element. If deviceTypes is not defined, the query results depend upon whether or not isServer is set and, if so, what that setting is. If isServer is set to true, the query results include only server devices. If isServer is set to false, the query results include only client devices. If neither deviceTypes nor isServer is set, query results include all three types of devices.

Endpoint

`<HOST>/flexnet/operations/manageDevice/devices`

Method

POST

URL Params

None

Data Params

None

Example

```
{"queryParams":
     {"alias":
          {"value":"[Optional. A string to match part or all of the alias value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "deviceId":
          {"value":"[Optional. A string to match part or all of the device ID]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "deviceIdType":
          {"value":"[A string to match part or all of the device ID Type value. For
                    example: UNKNOWN, STRING, ETHERNET, FLEXID9, FLEXID10, INTERNET, INTERNET6,
                    USER, TOLERANT, CONTAINER_ID, or VM_UUID.
                    When using a deviceType of SERVER, this field is mandatory.]",
          "searchType":"[One of the following: EQUALS or NOT_EQUALS]"
     },
     "parentId":
          {"value":"[Optional. A string to match part or all of the parentId which corresponds to the
                   Served By value for a served client in the admin console UI; it identifies the
                   served client’s parent FlexNet Embedded server device.]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "hostTypeName":
          {"value":"[Optional. A string to match part or all of the hostTypeName]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "soldTo":
          {"value":"[Optional. A string, in the form of the account ID, to match part or all of the
                   soldTo value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "soldToOrgId":
          {"value":"[Optional. A string, in the form of the account ID, to match part or all of the
                   soldToOrgID value.]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "description":
          {"value":"[Optional. A string to match part or all of the description value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "status":[The status of the device. One of the following: ACTIVE, OBSOLETE, or RETURNED],
     "addOnActivationId":
          {"value":"[Optional. A string to match part or all of the addOnActivationID value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "addOnProductName":
          {"value":"[Optional. A string to match part or all of the addOnProductName value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "addOnProductVersion":
          {"value":"[Optional. A string to match part or all of the addOnProductVersion value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "featureName":
          {"value":"[Optional. A string to match part or all of the featureName value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "isServer":[Optional. True if the device is a FlexNet Embedded server device, otherwise
                     False. See Note about isServer and device types.],
     "deviceTypes":[Optional. The deviceTypes element contains one or more WSDeviceType elements.
                    One of the following: CLIENT, SERVER, SERVED_CLIENT. See Note about isServer
                    and device types.],
     "userString":[Allows users to request devices with matching Device User. Results for this
                   request depend on how the tenant has configured Device User Option. If the Device
                   User Option is set to Off, the userString field is ignored, and the response
                   states that this criteria is not supported. If the Device User Option is set to
                   On with no validation, the userString is matched with string value of the Device
                   User. If the Device User Option is set to On with validation, the userString is
                   matched with the email of the Device User.],
     "deviceResponseConfig":
          {"alias":[true or false],
          "description":[true or false],
          "status":[true or false],
          "servedStatus":[true or false],
          "hostTypeName":[true or false],
          "soldTo":[true or false],
          "soldToOrgId":[true or false],
          "channelPartners":[true or false],
          "preBuiltLicense":[true or false],
          "addOnActivationId":[true or false],
          "addOnCounts":[true or false],
          "addOnProduct":[true or false],
          "addOnLicense":[true or false],
          "publisherIdentity":[true or false],
          "parent":[true or false],
          "machineType":[true or false],
          "vmDetails":[true or false],
          "vmInfo":[true or false],
          "vendorDictionary":[true or false],
          "deviceUser":[true or false]},
          "updates":[true or false]},
     "batchSize":[Required. Maximum number of records to return in the response. e.g. 100],
     "pageNumber":[Defaults to 1. For example: If query returns 100 records and the batch size is
                   set to 50, enter 2 to see the second page.]
}
```

Response

```
{
   "statusInfo":    {
      "status": "SUCCESS",
      "reason": null
   },
   "failedData": null,
   "responseData": {"device": [   {
      "deviceIdentifier":       {
         "deviceType": "SERVER",
         "uniqueId": null,
         "deviceId": "HV195HYU9F92",
         "serverIds": {"serverId":          [
            "HV195HYU9F92",
            null
         ]},
         "deviceIdType": "STRING",
         "publisherName": "fnedemo"
      },
      "alias": null,
      "description": null,
      "hostTypeName": null,
      "deviceStatus": null,
      "deviceServedStatus": null,
      "channelPartners": null,
      "soldTo":       {
         "displayName": "ACC-ANU-MY19-1",
         "id": "ACC-ANU-MY19-1",
         "name": "ACC-ANU-MY19-1"
      },
      "soldToOrgId":       {
         "displayName": "ACC-ANU-MY19-1",
         "id": "ACC-ANU-MY19-1",
         "name": "ACC-ANU-MY19-1"
      },
      "hasPrebuiltLicense": null,
      "prebuiltLicense": null,
      "hasAddonLicense": null,
      "addonLicense": null,
      "publisherIdName": null,
      "addonLineItemData": [],
      "featureData": null,
      "parentIdentifier": null,
      "machineType": null,
      "vmInfo": null,
      "updates": [
          {
              "updateId": "Notif4",
              "downloadPackageId": "12",
              "platform": "Windows",
              "language": "English"
           }
      ],
      "vendorDictionary": null,
      "deviceUser": null,
      "deviceUserIdentifier": null
   }]}
}
```

Devices Count

The Devices Count operation takes one or more query parameters to return the number of matching devices. If no specific query parameters are provided in the <queryParams> element, the total number of devices is returned.

Note ▪ *Some elements apply only to served client devices while others apply only to server devices and client devices. Elements that start with feature are specific to served clients. Elements that start with addOn or preBuilt are specific to server devices and client devices.*

Query results can be restricted to one or more device types listed in the deviceTypes element. If deviceTypes is not defined, the query results depend upon whether or not isServer is set and, if so, what that setting is. If isServer is set to true, the query results include only server devices. If isServer is set to false, the query results include only client devices. If neither deviceTypes nor isServer is set, query results include all three types of devices.

Endpoint

<*HOST*>/flexnet/operations/manageDevice/devicesCount

Method

POST

URL Params

None

Data Params

None

Example

```
{"queryParams":
     {"alias":
          {"value":"[Optional. A string to match part or all of the alias value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "deviceId":
          {"value":"[Optional. A string to match part or all of the device ID]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "deviceIdType":
          {"value":"[Optional. A string to match part or all of the device ID Type value. For
                     example: UNKNOWN, STRING, ETHERNET, FLEXID9, FLEXID10, INTERNET, INTERNET6,
                     USER, TOLERANT, CONTAINER_ID, or VM_UUID]",
          "searchType":"[One of the following: EQUALS or NOT_EQUALS]"
     },
     "parentId":
          {"value":"[Optional. A string to match part or all of the parentId which corresponds to the
                    Served By value for a served client in the admin console UI; it identifies the
                    served client’s parent FlexNet Embedded server device.]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "hostTypeName":
          {"value":"[Optional. A string to match part or all of the hostTypeName]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "soldTo":
          {"value":"[Optional. A string, in the form of the account ID, to match part or all of the
                    soldTo value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "soldToOrgId":
          {"value":"[Optional. A string, in the form of the account ID, to match part or all of the
                    soldToOrgID value.]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
     },
     "description":
          {"value":"[Optional. A string to match part or all of the description value]",
          "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
```

```
        },
        "status":[The status of the device. One of the following: ACTIVE, OBSOLETE, or RETURNED],
        "addOnActivationId":
             {"value":"[Optional. A string to match part or all of the addOnActivationID value]",
             "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
        "addOnProductName":
             {"value":"[Optional. A string to match part or all of the addOnProductName value]",
             "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
        "addOnProductVersion":
             {"value":"[Optional. A string to match part or all of the addOnProductVersion value]",
             "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
        "featureName":
             {"value":"[Optional. A string to match part or all of the featureName value]",
             "searchType":"[One of the following: STARTS_WITH, CONTAINS, ENDS_WITH, or EQUALS]"
        },
        "isServer":[Optional. True if the device is a FlexNet Embedded server device, otherwise
                          False. See Note about isServer and device types.],
        "deviceTypes":[Optional. The deviceTypes element contains one or more WSDeviceType elements.
                       One of the following: CLIENT, SERVER, SERVED_CLIENT. See Note about isServer
                       and device types.],
        "userString":[Allows users to request devices with matching Device User. Results for this
                       request depend on how the tenant has configured Device User Option. If the Device
                      User Option is set to Off, the userString field is ignored, and the response
                      states that this criteria is not supported. If the Device User Option is set to
                      On with no validation, the userString is matched with string value of the Device
                      User. If the Device User Option is set to On with validation, the userString is
                      matched with the email of the Device User.]
        }
}
```

Response

```
{
   "statusInfo":    {
      "status": "SUCCESS",
     "reason": null
   },
   "failedData": null,
   "responseData": {"count": 1}
}
```

Consumption

The consumption REST API provides feature consumption information for a given account ID.

Endpoint

`<HOST>/flexnet/operations/consumption`

Method

GET

URL Params

accountID=[*alphanumeric*]

Data Params

None

Example

`<HOST>/flexnet/operations/health/consumption?accountId=ACME`

Response

```
{
   "statusInfo":    {
      "status": "SUCCESS",
      "reason": null
   },
   "consumption":    [
            {
         "accountId": "ACME",
         "activationId": "ACT-ceda-610B-0E39-f41",
         "productName": "pP_030915_2",
         "productVersion": "1.0",
         "featureName": "",
         "featureVersion": "",
         "consumed": 0,
         "total": 50,
      },
            {
         "accountId": "ACME",
         "activationId": "E-SEP303-LNM1",
         "productName": "FC_1",
         "productVersion": "1.0",
         "featureName": "",
         "featureVersion": "",
         "consumed": 37,
         "total": 100,
      },
            {
         "accountId": "ACME",
         "activationId": "4322-b6af-cd28-4415-9a33-92ef-7e53-0997",
         "productName": "Pt-Anu-AutoUpgrade-1",
         "productVersion": "1.0",
         "featureName": "",
         "featureVersion": "",
         "consumed": 0,
         "total": 123,
      },
            {
         "accountId": "ACME",
         "activationId": "PFEBFC-bb53-8577-DD9f-251",
         "productName": "P-FEB191",
         "productVersion": "1.0",
         "featureName": "",
         "featureVersion": "",
         "consumed": 0,
         "total": 100,
      },
.
.
.
            {
         "accountId": "ACME",
         "activationId": "ACT-725a-fe79-4241-551",
         "productName": "18353_custom_1",
         "productVersion": "1",
         "featureName": "",
         "featureVersion": "",
         "consumed": 0,
         "total": 11,
      }
   ]
}
```

6

External Web Services

Producers can create external services to customize aspects of FlexNet Operations behavior, and Revenera provides sample code and other resources to facilitate these customization efforts.

The ways in which these external services can be implemented depends on how the FlexNet Operations component to be customized is exposed to be thus extended or altered. Some components can be customized via SOAP-based web services, others by REST-based web services, and some are only customizable via Java classes added to the class path on the machines that host FlexNet Operations.

This chapter covers the components that are customizable by users.

Table 6-1 ▪ External Web Services Information

Topic	Description
Sample Source and Other Resources	Describes how producers can obtain sample code and other resources for implementing external services.
Using External Services	Provides an overview of how FlexNet Operations interacts with external services.
Customizable Features	Describes the interfaces that allow producers to customize FlexNet Operations behavior. ● Renewals ● ID Generator ● License Generator ● Capability Request Callout

Tip ▪ *Additional information about customizing FlexNet Operations via custom Java code appears in the FlexNet Operations On Premises Installation & Implementation Guide.*

Sample Source and Other Resources

Sample source code and other resources are available from the Product and License Center and from within the FlexNet Operations install directory as well. There are two archives that contain samples:

- `extgenservice.zip` contains sample source for most customizable FlexNet Operations core components, like license generation, filename generation, ID generation, and renewal request handling.
- `flexnet-lfs-callout.war` contains sample source for customizing the handling of capability requests and responses.

Obtaining External Service Samples

The sample files are provided under the following directory:

`<ops_install_dir>\release\flexnet-data\site\samples\`

- You can find `extgenservice.zip` in `<ops_install_dir>\release\flexnet-data\site\samples\externalservices\`.
- You can extract `flexnet-lfs-callout.war` by running the extract target in the build script in `<ops_install_dir>\release\flexnet-data\site\samples\lfs\capabilityrequest\`.

Note that these files (`extgenservice.zip` and `flexnet-lfs-callout.war`) are also available from the Product and License Center in the same location where you download the FlexNet Operations installer.

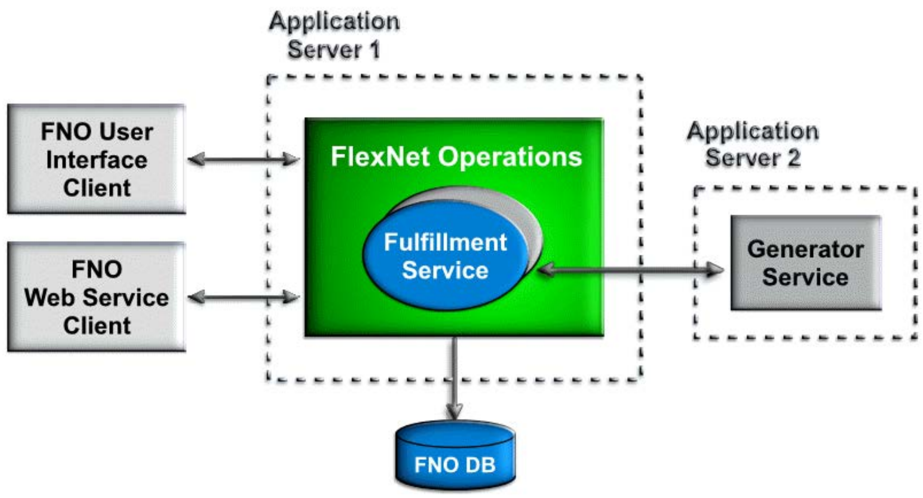
You will find additional sample files included in your samples directory, which provide on-premises-specific implementations.

Using External Services

Producers can use Revenera-provided interfaces to deploy external services that provide customized functionality within FlexNet Operations. For example,

- License keys can be generated by an external SOAP service that implements the **LicenseGeneratorService** WSDL.
- Custom IDs can be generated by an external SOAP service that implements the **IdGeneratorService** WSDL.
- Custom handling of renewal requests can be supported using an external SOAP service that implements the RenewalService WSDL.
- Capability request handling can be customized through an external REST service that processes JSON requests and produces JSON responses that dictate what actions to perform.

The server that implements an external service is hosted separately from the one that runs FlexNet Operations. It operates independently of FlexNet Operations, as shown in the figure below. The external service is accessed by FlexNet Operations at run time for operations such as capability request handling, ID generation, and license generation.



Note ▪ *The availability of certain operations for customization varies with your FlexNet Operations license and other factors. This guide covers the operations producers can customize via web services. For information on other customization options, consult the FlexNet Operations On Premises Installation & Implementation Guide.*

Customizable Features

The following customizable features are described here:

- Renewals
- ID Generator
- License Generator
- Capability Request Callout

Renewals

Two samples are provided in `extgenservice.zip`:

- `RenewalServiceImpl.java` – Shows how to access values from input parameters if the producer needs to make an entry into their own system.
- `RenewalRedirectServiceImpl.java` – Shows how to access values from input parameters if the producer needs to use an external application/portal with few parameters from FlexNet Operations.

These samples are contained in `extgenservice.zip` at

`extgenservice\src\com\flexnet\external\webservice\renewal`

Using code like that shown in these samples enables a producer to configure their End-User Portal to directly send renewal requests to their back-office system for processing.

When you create a SOAP endpoint that implements the **RenewalService** interface, specify the endpoint location in the FlexNet Operations system configuration settings. In the Producer Portal, click **System** > **Configure** > **FlexNet Operations** > **Renewals** and then type the location in **Renewal Callout**. The End-User Portal uses this endpoint when an end user clicks the **Request** button on the End-User Portal's **Expiring Entitlements** page.

For more information on renewal management, see the , sections Getting Started with FlexNet Customer Growth and Configuring FlexNet Operations.

You can find additional information and instructions for implementing a renewal request handler callout as a custom Java class in the *FlexNet Operations On Premises Installation & Implementation Guide*.

ID Generator

Customized IDs can be implemented using an ID generator webapp hosted in Tomcat in the cloud. The generation of following types of IDs can be customized:

- Entitlement IDs
- Line Item IDs
- Web Register Keys
- Maintenance Line Item IDs
- Fulfillment IDs
- Consolidated License IDs

A sample class, `IdGeneratorServiceImpl.java`, can be found in `extgenservice.zip` in

`extgenservice\src\com\flexnet\external\webservice\idgenerator`

When you create a SOAP endpoint that implements the **IdGenerator** interface, specify the endpoint location in the FlexNet Operations system configuration settings. In the Producer Portal, click **System** > **Configure** > **FlexNet Operations** > **General Options** and then type the location in **ID Generator Classname** or **Fulfillment ID Generator Classname**.

See the *FlexNet Operations On Premises Installation & Implementation Guide* for more information on customizing features with Java classes.

License Generator

The license generator service includes support for validating a product, validating a license model, generating a license, consolidating fulfillments, generating license file names, generating consolidated license file names, and generating custom host IDs.

Sample implementations are in `LicenseGeneratorServiceImpl.java` in `extgenservice.zip` in

`extgenservice\src\com\flexnet\external\webservice\keygenerator`

See Using External Services with License Technologies in the section Administrator Reference in the for more information.

Capability Request Callout

Customized handling of capability requests can be implemented using a RESTful external service.

A sample implementation can be downloaded from the Product & License Center or extracted from a FlexNet Operations installation as `flexnet-lfs-callout.war`.

There are several trigger points during the capability request handling at which you can apply custom behavior. See Actions for a list of possible customizations.

Usage

Always use standard ways of configuring the license before using this callout. It might be tempting to put a lot of functionality in this callout. However, since all this logic is in the callout code, there is no way to analyze its workings through Producer Portal. This can hamper Revenera support and engineering's ability to analyze problems.

Avoid unnecessary web service calls during capability request handling. The callout is running remotely (meaning on a server other than the FlexNet Operations application) and is called in the middle of the capability request handling. Although it is possible to call FlexNet Operations public web services from the callout (or public web services hosted anywhere else), the performance impact on capability request processing would be significant and is strongly discouraged.

Trigger Points

The following trigger points can be enabled. When enabled, the trigger points allow FlexNet Operations to make a callout in which you can code additional functionality:

- **Finalize host**: just before creating an unknown host
- **Check access**: before proceeding with processing an off-line request
- **Finalize response**: after an incoming host request has been validated, but before the capability response has been generated

The trigger point being invoked is encoded in the URL to which the JSON is posted. These path parameters are appended to the configured URL to indicate which trigger point is active: `/finalizeHost`, `/checkAccess`, or `/finalizeResponse`. You can look at the example servlet class to see how this is used when invoking the callout.

Actions

The capability request callout supports a number of different actions. Use only actions within their approved trigger points. Although it would be possible to call FlexNet Operations public web services from the callout to take any action that the web services API supports, the performance impact on capability request processing can be significant; hence, the use of public web services at any trigger point is strongly discouraged.

The following table lists the kinds of custom actions that producers can implement and the trigger points at which the action can be applied.

Table 6-2 ▪ Actions and trigger points

Action	Possible Trigger Points
Set the capability response lifetime	Finalize response only

Table 6-2 ▪ Actions and trigger points

Action	Possible Trigger Points
Add status list items to a capability response	All: Finalize host, check access, and finalize response
Add vendor dictionary to a capability response	Finalize response only
Add vendor dictionary entries to a target host instance	Finalize host and finalize response
Remove vendor dictionary entries from a target host instance	Finalize host and finalize response
Deny the creation of an unknown host	Finalize host only
Deny access to a target host (return an error instead)	Check access only
Set the host type of a target host	Finalize host only
Set an owner (enterprise ID) of a target host	Finalize host and finalize response
Skip confirmation when reducing add-on copies	Finalize response only
Deny mapping of an add-on or inclusion of a mapped add-on in a generated license	Finalize response only
Set an add-on mapping expiration date override	Finalize response only

Sample Implementation

The following notes provide guidance on how to implement the capability request callout.

Since the callout mechanism is fairly complex, and no official API is provided to aid in implementation, Revenera includes a sample Java implementation with FlexNet Operations resources: `flexnet-lfs-callout.war`. This WAR file includes a servlet suitable for deployment in a servlet container like Tomcat. The sample code shows one way to parse the request and prepare the response (using Jackson). It demonstrates some of the most common actions such as adding vendor dictionary entries to the response.

***Tip* ▪** *Instructions for obtaining `flexnet-lfs-callout.war` are in Sample Source and Other Resources.*

Producers can either host their own external REST callout implementation in Tomcat or they can run the callout in the LFS classpath (packaging the compiled callout back in to the `lfs.war`).

The `build.xml` in `<ops_install_dir>\release\flexnet-data\site\samples\lfs\capabilityrequest\` is used for both implementation options. The `build.xml` file has comments to explain the build targets and how to use them. The *extract* target is used to extract sample source and dependencies from `lfs.war` in the installation. The extract target is intended for producers that want to take the class path approach (as opposed to an external service approach).

For producers using the servlet approach, just run the extract-all target of that `build.xml` file to get the sample files extracted from `lfs.war`.

Related System Configurations

Because the callout will, in many cases, be running remotely without direct access to the database, it must be passed all the information needed by the callout logic up front. As this could be quite a lot of data, and will happen for every request, the performance penalty could be significant. To allow customers to minimize the impact, a number of configuration settings control when the callout is invoked, and what data will be include in the request body sent to the callout.

The related configurations are in the section: **System** > **Configure** > **Embedded Devices** > **Capability Request Handling**.

Callout Location

The system configuration setting, **Capability Request Callout**, sets location where FlexNet Operations can find the callout:

- a URL for an external service implementing the callout
- a fully qualified class name of a class added to the LFS class path that implements the callout.

If the value starts with http:// or https:// the setting is interpreted as a URL, otherwise it is assumed to be a class name.

Enabling Trigger Points

The following system configuration settings determine which trigger points are active.

- **Capability Request Finalize Host Callout**—If true, invoke callout just before creating an unknown host. Additional attributes may be defined before it is persisted. It can also be used to cancel creation of the new host instance.
- **Capability Request Access Check Callout**—If true, invoke callout after (optionally) creating an unknown host, before proceeding with processing an off-line request. The return value can be used to deny access to the targeted host.
- **Capability Request Finalize Response Callout**—If true, invoke callout after an incoming host request has been validated, but before the capability response has been generated.

See Trigger Points for more information about trigger points. See Actions to find out which actions can be implemented at each trigger point.

Data to Send in the Request

The following system configuration settings determine how much data is included in the capability request.

- **Callout Includes Host Details**—If true, send details about the target host in the callout request body, otherwise just the host id and id type, host class, host type and alias.
- **Callout Includes Add-on Details**—If true, send add-on details in the callout request body, otherwise just send list of activation IDs.

JSON Reference

The following examples show the syntax of the request body and response body.

JSON Request Body

The JSON message POSTed to the callout URL (or passed to the invoke() method if class name used) contains several top-level fields.

Field	Description
requestInfo	contains information from the capability request.
hostInfo	contains information from the target host of the capability request; the attributes in *italics* are only included if the system configuration setting, **Callout Includes Host Details** (capabilityService.callout.includeHostDetails) is enabled.
addOnInfo	contains information about the add-ons mapped to the host; the attributes in *italics* are only included if the system configuration setting, **Callout Includes Add-on Details** (capabilityService.callout.includeAddOnDetails) is enabled.
statusList	contains status list items that are pending for including in the capability response.

Below is a sample request body:

```
{
    "requestInfo" : {

        "activationIds" : [
            { "id" : "a1", "copies" : 1, "partial" : false },
            { "id" : "a2", "copies" : 1000, "partial" : true }
        ],
        "vendorDictionary" : {
            "somekey" : 10,
            "anotherkey" : "anothervalue"
        },
        "lastResponseTime", 1502906444,
    }

    "hostInfo" : {
        "id" : "dev1",
        "idType" : "STRING",
        "hostClass" : "CLIENT",
        "hostType" : "FLX_CLIENT",
        "alias" : "My device",
        "identityName" : "id1",
        "publisherName" : "demo",
        "status" : "ACTIVE",
        "vendorDictionary" : {
           "somekey" : 10,
            "anotherkey" : "anothervalue"
        },
        "firstActivated" : "2017-08-09T12:34:56Z",
        "machineType" : "PHYSICAL",
        "vmName" : "Amazon EC2",
        "baseProductId" : "baseprod1",
         "enterpriseId" : 101,
        "user" : "joe@blow.com",
        "userId" : 19
    },

    "addOnInfo" : [
        {
            "activationId" : "a1",
            "requested" : 1,
            "consumed" : 2,
             "waitingForConfirmation" : true,
             "expirationOveride" : null,
             "licenseModel" : "Embedded Counted",
             "vendorString" : "some vendor string",
             "notice" : null,
             "serialNumber" : null,
             "issuer" : null,
        }, ...
    ],

    "statusList" : [
        {
            "code" : 1,
            "detail" : "a9"
        }, ...
    ]
}
```

JSON Response Body

The JSON message returned from the callout can contain several top level fields. `responseActions` defines action to apply to the capability response such as adding vendor dictionary entries, changing the status list entries, or the response lifetime. `hostActions` defines actions to apply to the target host instance, such as modifying the vendor dictionary entries, denying host creation, and setting the host type or the host owner. `addOnActions` defines actions to apply to the add-ons mapped to the host, such as skipping removal confirmation, denying an add-on from being mapped or included in the response, and overriding the expiration date.

Here is a sample response body:

```
{
    "responseActions" : {
        "vendorDictionary" : {
            "somekey" : 1,
            "anotherkey" : "anothervalue",
        },
        "addToStatusList" : [ { "code" : 5, "detail" : "a2" }, ... ],
        "lifetime" : 0
    },

    "hostActions" : {
        "addToVendorDictionary" : { "key" : "value", ... },
        "removeFromVendorDictionary" : [ "key", ... ],
        "denyCreate" : true,
        "hostType" : "CustomHostType",
        "enterpriseId" : 20
    },

    "addOnActions" : [
        {
            "activationId" : "a1",
            "skipConfirmation" : true,
            "denied" : true,
            "expiration" : null
        }, ...
    ]
}
```